175 PTAS



86

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR



MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VIII-Fascículo 86

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-067-2 (tomo 7)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 188509
Impreso en España-Printed in Spain-Septiembre 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

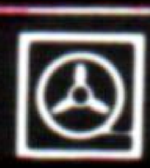
c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

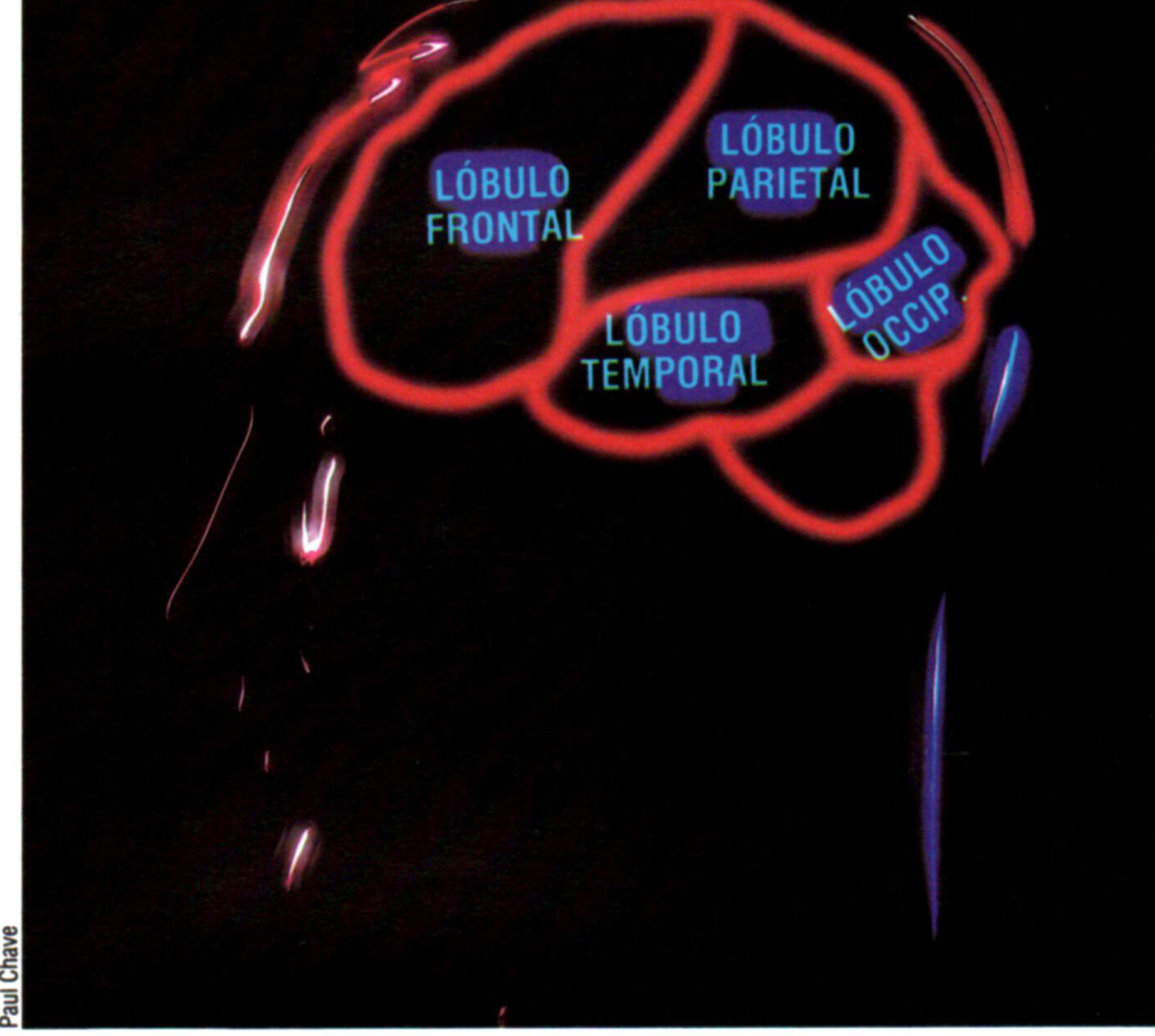
Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.





En el cerebro



Paul Chave

Iniciamos una serie en la que trataremos en profundidad el apasionante tema de la inteligencia artificial y los importantes avances de la investigación en este campo

A principios de la década de los setenta, las investigaciones en el campo de la inteligencia artificial (*artificial intelligence*: AI) se hallaban estancadas. Se la consideraba, en líneas generales, como un aspecto marginal y extravagante de la ciencia informática. Por aquel entonces, en Gran Bretaña, un elaborado informe de sir James Lighthill para el Science Research Council recomendaba un drástico recorte de los fondos destinados a tal fin. En la actualidad, todo lo concerniente a la inteligencia artificial se encuentra en pleno auge, y los profesionales dedicados a su investigación se ven acosados por audaces inversionistas que les proponen tentadoras ofertas económicas.

Varias agencias gubernamentales europeas están financiando costosos programas de investigación y desarrollo por temor a quedarse rezagadas en la carrera por el desarrollo de la AI. Mientras, las firmas vendedoras de software están distribuyendo pomposos comunicados de prensa en los que redefinen sus productos como sistemas de inteligencia artificial.

Para comprender la situación en que se encuentra hoy la AI y la que es probable que alcance en el futuro, es útil, tal como sucede con tantas otras tecnologías, dar una mirada a su pasado. Podemos dividir nuestra condensada historia de la inteligencia artificial en cuatro períodos, cada uno de ellos de una década de duración y caracterizado por un tema dominante. Esto necesariamente supone una simplificación de las cosas, pero al hacerlo, se ponen de relieve los puntos principales. Podemos considerar que cada uno de los temas esenciales es la respuesta que con toda probabilidad habría obtenido uno de haberle preguntado a un investigador de AI de la época: "¿En qué consiste la inteligencia artificial?"

1950 Redes neurales
1960 Búsqueda heurística
1970 Sistemas expertos
1980 Aprendizaje de la máquina

En 1943, Warren McCulloch y Walter Pitts propusieron un modelo de neurona del cerebro humano y animal. Estas células nerviosas abstractas proporcionaron la base para una representación simbólica matemática de la actividad cerebral. Otros investigadores, en especial Norbert Wiener, elaboraron estas ideas junto con otras similares, dentro del mismo campo, que se dio en llamar "cibernética" (un sistema basado en la premisa de que se puede construir una máquina consciente utilizando como modelo métodos biológicos de realimentación y análisis). De la cibernética surgió, en la década de los cincuenta, la inteligencia artificial.

Los primeros investigadores de AI tomaron como su bloque constructivo la neurona formalizada de McCulloch. Considerando la inmensa complejidad del cerebro, no fue del todo sorprendente que no consiguieran generar sistemas inteligentes basados en este modelo. En efecto, ellos postulaban: "El cerebro es un solucionador inteligente de problemas, de modo que imitemos al cerebro." Pero el hardware de la época, por no hablar del software, no estaba a la altura de la tarea.

Uno de los pocos sistemas que obtuvo cierto éxito en aquellos días fue el Perceptron de Rosenblatt. Se trataba de un sistema visual elemental al cual se le podía enseñar a reconocer patrones. Como se puede apreciar en el diagrama (p. 1703), el Perceptron se compone de una cuadrícula finita de células sensibles a la luz que conforman una retina en miniatura. Además hay varios elementos detectores de configuración (denominados gráfica-

Mentes mecánicas
La inteligencia artificial trata del diseño de sistemas de ordenador que lleven a cabo tareas que, de realizarlas un ser humano, requerirían inteligencia. Sin embargo, su esfera de acción se ha ampliado para incluir funciones de percepción, tales como ver y oír. La principal preocupación de la investigación sobre AI es programar máquinas de modo que imiten aspectos del comportamiento y la comprensión humanos

mente *demonios*) que controlan el estado de grupos de células de la cuadrícula. Cuando hay presentes subpatrones característicos, responden enviándole una señal a un elaborador de decisiones. Éste multiplica cada señal proveniente de un demonio local por un factor de valor relativo positivo o negativo, y se suman los números resultantes. Si el total supera un umbral establecido, un Perceptron puede distinguir entre dos clases de imágenes, si bien se pueden ampliar los mismos principios para tratar con más de dos.

En un momento dado hubo grandes esperanzas de que el Perceptron pudiera abordar una amplia gama de tareas de resolución de problemas, pero éstas se diluyeron enseguida. Los estudiosos de la AI comenzaron a considerar el pensamiento humano como una coordinación de tareas esencialmente simples relacionadas con la manipulación de símbolos. Si bien esto constituyó ostensiblemente un gran cambio en la dirección de las investigaciones, los diseñadores se hallaban al menos en terreno firme, puesto que los ordenadores podían hacer cosas tales como efectuar búsquedas, comparar símbolos, etc., que ellos identificaban con los fundamentos de la solución inteligente de problemas. Lo difícil consistía en unir entre sí estas actividades simples.

Los estudiosos más influyentes de la década de los sesenta fueron Alan Newell y Herbert Simon, de la Universidad de Carnegie-Mellon, que, entre otras investigaciones, trabajaron en demostración de teoremas y ajedrez por ordenador. Su logro más impresionante fue un programa denominado GPS (*General Problem Solver*: solucionador general de problemas). Éste era general en el sentido de que el usuario definía un "entorno de tarea" en función de los objetos de un dominio particular y los operadores que se podían aplicar a esos objetos. Sin embargo, esta generalidad se ceñía a puzzles con un juego relativamente pequeño de estados y reglas bien definidos. Podía trabajar con las torres de Hanoi, criptoaritmética y otros tipos de problemas similares en los que microcosmos formalizados representaban los parámetros dentro de los cuales se podían realmente resolver problemas. Lo que no podía hacer el GPS era resolver lo que la gente consideraría problemas del mundo real, como realizar diagnósticos médicos o tomar decisiones de gestión basadas en monedas fluctuantes.

El GPS se sustentaba en la idea de que la solución del problema implicaba una búsqueda a través de un espacio de soluciones potenciales. Para que la búsqueda fuera eficaz debía estar dirigida por reglas heurísticas (que aprenden a partir de sus propios descubrimientos) que la conducían hasta el destino deseado, hallando el camino esencialmente a través del método de ensayo y error. Por consiguiente, un robot que deambulara a través de un laberinto tendría que utilizar una técnica de búsqueda exhaustiva si no supiera nada sobre la estructura de ese laberinto; pero si tuviera alguna forma de saber cuándo se estaba "acercando", se podía esperar que llegara a su estado objetivo más rápidamente (pero no siempre, porque no existen garantías de que la heurística funcione, y en ocasiones puede conducir a callejones sin salida). Durante este período, los estudiosos de la AI idearon varias estrategias de búsqueda guiadas heurísticamente.

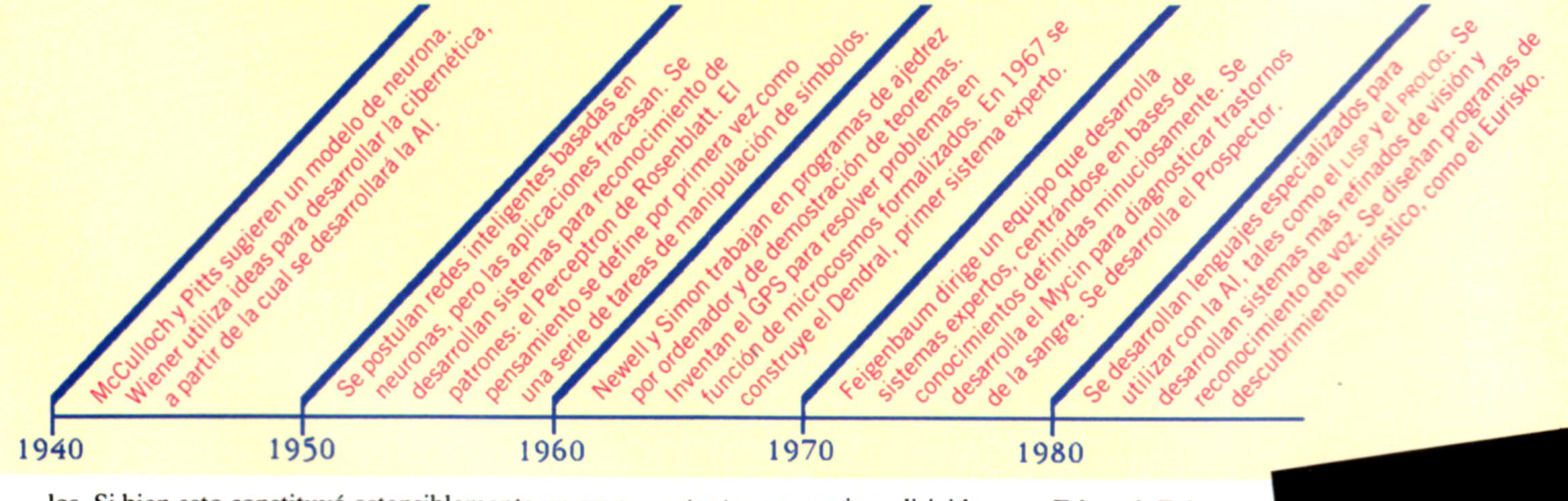
El GPS, como hemos dicho, no era muy eficaz para resolver problemas de la vida real. En los años setenta un equipo dirigido por Edward Feigenbaum, de la Universidad de Stanford, comenzó a remediar ese defecto. En vez de intentar informatizar la inteligencia general, se centraron en áreas muy concretas de pericia. Y así nació el sistema experto.

El primer sistema experto fue el Dendral, un intérprete de espectrograma de masa construido ya en 1967, si bien el más influyente resultaría ser el Mycin, que data de 1974. El Mycin diagnostica infecciones bacterianas de la sangre y receta la medicación terapéutica. Ha dado lugar a toda una familia de "clones" para diagnóstico médico, algunos de los cuales son de uso clínico común. Por ejemplo, Puff, una herramienta para diagnóstico de función pulmonar basada en el plan Mycin, es de uso común en el Pacific Medical Center, situado en las cercanías de San Francisco.

El Mycin introdujo varias características nuevas que se han convertido en los rasgos distintivos de los sistemas expertos. En primer lugar, su "conocimiento" consiste en cientos de reglas como ésta:

REGLA N.º 47:
IF
1) la ubicación del cultivo es sangre, y
2) la identidad del organismo no se conoce con certeza, y
3) la tinción del organismo es gramnegativa, y
4) la morfología del organismo es de bastoncito, y
5) el paciente ha sufrido graves quemaduras

Perspectiva histórica
La inteligencia artificial se está convirtiendo en un área de investigación práctica cada vez más importante, con aplicaciones en muchos campos. La historia de la AI, sin embargo, abarca un período muy breve en comparación con muchos otros campos científicos, como se puede apreciar en el diagrama inferior...



Liz Dixon


Cortesía de Computer Weekly

Edward Feigenbaum
Como director de un equipo de AI de la Universidad de Stanford (California), Edward Feigenbaum desarrolló los primeros sistemas expertos. Éstos fueron notables porque, apartándose de la idea de programas de inteligencia general, se constituyeron en sistemas que operaban dentro de límites de conocimientos y objetivos definidos con suma precisión



THEN existen evidencias que sugieren débilmente (0,4) que la identidad del organismo es seudomomas.

En segundo lugar, estas reglas son probabilísticas. Shortliffe, el inventor del Mycin, que era doctor en medicina, desarrolló un esquema basado en factores de certidumbre para permitir que el sistema llegara a conclusiones verosímiles a partir de una evidencia incierta. Por consiguiente, el número 0,4 no es estrictamente una probabilidad; es, esencialmente, un "factor de falsedad". El punto significativo es, sin embargo, que el Mycin y sistemas similares pueden llegar a conclusiones correctas aun con información incompleta y parcialmente errónea. Emplean un método de razonamiento aproximado (basado ya sea en probabilidades, *fuzzy logic*, factores de certidumbre o algún otro cálculo de probabilidad) para producir una buena estimación de la verdad a partir de un conjunto imperfecto de datos.

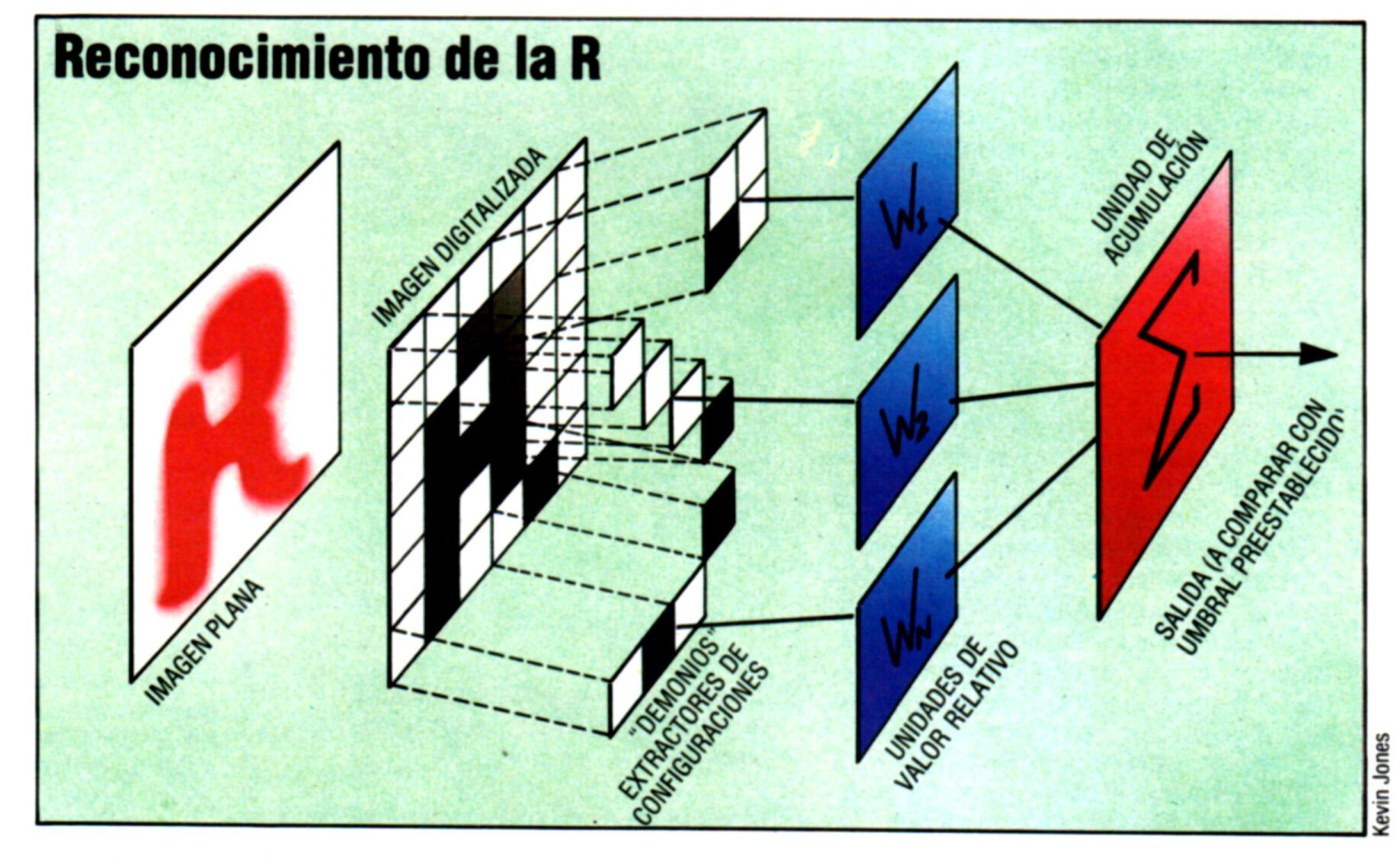
En tercer lugar, el Mycin puede explicar su propio proceso de razonamiento. El médico que lo utiliza puede interrogarlo de diversas maneras, ya sea para preguntarle cómo llegó a una determinada conclusión o bien por qué está solicitando una cierta clase de información. El sistema responde desandando y describiendo el proceso deductivo que condujo al estado actual. Este nivel de amabilidad hacia el usuario fue esencialmente un derivado del estilo de programación basado en reglas.

El factor último y esencial es que el Mycin *funciona*. Ejecuta aquello que un ser humano sólo puede hacer tras años de entrenamiento. En realidad, el Mycin se utiliza más para la enseñanza que para el diagnóstico, pero lo cierto es que está despertando un creciente interés entre las grandes corporaciones, los gobiernos y los *mass media*.

Y así llegamos a los años ochenta. Los sistemas expertos están de moda y su "ingrediente activo" es el conocimiento, porque el alcance y la calidad de su base de conocimientos determina el éxito de un sistema experto. Pero el conocimiento no es algo que uno pueda comprimir en un programa de ordenador como si se tratara de la pasta dentífrica de un tubo. Codificar la pericia de un experto humano puede ser un proceso largo y laborioso. De modo que mientras el mundo se maravilla ante los sistemas expertos, la inteligencia artificial ha pasado a concentrarse en el problema del aprendizaje de la máquina, que es una forma de adquirir conocimiento de forma automática. La AI siempre ha sido un objetivo móvil y ahora mismo en el centro de ese objetivo hay un programa que se denomina Eurisko. Se trata de un programa de descubrimiento que amplía y perfecciona su propio cuerpo de reglas heurísticas automáticamente, por inducción. Aparte de ganar durante tres años consecutivos el juego de guerra naval *Trillion credit squadron* (a pesar de que se intentó impedirlo mediante cambios en el reglamento), el Eurisko también se ha aplicado a problemas prácticos. Un resultado fue la invención de una nueva puerta lógica tridimensional en el campo del diseño de circuitos integrados. Existen pocas dudas en el sentido de que sistemas como el Eurisko representan la vanguardia de la investigación en AI. Y puesto que la inteligencia artificial constituye la vanguardia de la ciencia informática, es aquí donde se han de buscar los indicios para el futuro de ésta.

No deja de ser irónico que, al volver a concentrarse en el aprendizaje, las investigaciones en el campo de la inteligencia artificial hayan regresado a sus raíces, porque en los primeros días de la cibernética el aprendizaje se consideraba el problema clave.

Esta serie de capítulos representa una guía práctica a la inteligencia artificial. A lo largo de la misma iremos cubriendo las principales áreas problemáticas de la AI (como el proceso de la visión y del lenguaje natural), así como las técnicas que se han desarrollado para abordarlas. Como es habitual, todos los programas que se ofrezcan a modo de ejemplo estarán en BASIC, y en los recuadros de *Complementos al* BASIC nos referiremos al BBC Micro, el Commodore 64 y el Spectrum.



Las percepciones del Perceptron
La imagen a muestrear (en este caso, la letra R) se proyecta sobre un plano y se digitaliza. Los *demonios* muestran un pequeño grupo de pixels (p. ej., 4 por vez) y responden si está presente el patrón para cuyo reconocimiento han sido programados. La respuesta de cada demonio (0 o 1) se multiplica por un factor de valor relativo, según cuál sea su importancia dentro del patrón como un todo, y se suman entre sí las respuestas. Este resultado se compara entonces con un valor umbral. Si es mayor que el umbral, entonces el sistema reconoce la forma original; de lo contrario, no la reconoce. El umbral y los valores relativos de los demonios se pueden regular cuando se entrena al Perceptron para que reconozca una forma determinada

La clasificación adecuada

Al ir de tiendas en busca de un administrador de bases de datos, ¿cuáles son los puntos principales que hay que tener en cuenta?

Cuando se utiliza un procesador de textos, el tiempo que consume la búsqueda de archivos o el desplazamiento de bloques de caracteres (lo que equivale a la clasificación) es un factor enojoso. Sin embargo, tras haberlas llevado a cabo, estas actividades por lo general no vuelven a ser necesarias. El esfuerzo de "búsqueda y clasificación" es, por tanto, un porcentaje relativamente pequeño del tiempo total de ejecución del programa. Es la entrada de datos (digitar el texto) y la impresión lo que tienden a consumir la mayor cantidad de tiempo.

En el caso de los DBM, sin embargo, esta situación se invierte. En términos generales, los registros son "documentos" muy pequeños cuya entrada demanda poco tiempo. Cuando se utiliza un DBM, éste invierte la mayor parte de su tiempo clasificando los registros y buscando a través de ellos para extraer los que se requieren. Si el archivo de datos es mayor que la capacidad total de la RAM, como es lo más probable, se requerirá una exhaustiva lectura y escritura en un soporte de almacenamiento masivo. Las deficiencias del almacenamiento en cassette, lento y secuencial, se harán evidentes de inmediato. A menos que su base de datos sea limitada en tamaño y en importancia (el catálogo de una pequeña colección de discos, p. ej.), usted deberá considerar el tener que invertir en un sistema basado en disco.

La mayoría de las bases de datos se emplean en un contexto de gestión de uno u otro tipo, en los que rige el lema *time is money* (el tiempo es dinero). En otras palabras, si necesita un DBM, precisará un sistema basado en disco, aunque, mientras tanto, el *Archive* en los microdrives del QL constituye una alternativa adecuada.

¿Cuántos registros ha de ser capaz de manipular mi DBM? ¿Vale la pena tener uno con más registros de los que realmente necesito?

Su aplicación de base de datos puede o no exigir la capacidad de manipular gran número de registros. Pero intente calcular el número máximo que podrá necesitar alguna vez, duplíquelo, y después busque un DBM capaz de manipular esa cantidad. Si su base de datos es para control de stock e inventario y no cuenta con muchos componentes, puede que no necesite un DBM capaz de manipular 32 000 registros. Pero si está utilizando el DBM para catalogar una biblioteca, quizá 32 000 registros no sean suficientes. Los DBM capaces de manejar más de 64 000 registros son bastante raros y, de todos modos, normalmente requerirían potentes ordenadores con memorias muy grandes en discos rígidos.

Si no estoy muy seguro respeto a la cantidad de campos que necesitaré, ¿cómo puedo decidir qué DBM comprar?

La mayoría de los DBM permiten que cada registro contenga sólo una cierta cantidad de campos. A menudo resulta difícil saber de antemano cuántos campos se requerirán. La mayoría permiten un número de campos más que suficiente, si bien la longitud de éstos a menudo se limita a una línea, por lo cual puede resultar complicada la utilización de campos de textos de más de una línea. Una limitación más común que la cantidad de campos permitida es el número de caracteres admisibles por registro. Una cifra típica podría ser 1 020 caracteres por registro, permitiendo al usuario elegir cómo dividir esta cantidad entre el número de campos y la longitud de caracteres de cada campo. En cualquier caso, asegúrese de que el número máximo de caracteres por campo y de campos por registro permita que usted construya los registros que desea.

Los campos clave ¿son un aspecto importante? De ser así, ¿cuántos necesitaré?

Un campo clave es aquel mediante el cual el DBM puede efectuar una búsqueda u otra manipulación. Si un campo no está designado como clave, no puede ser utilizado para extraer registros. He aquí dos ejemplos de cómo emplear los campos clave típicos:

```
SEARCH PROVEEDOR FOR 'Timson Engineering Ltd'
```

Se está buscando en el campo clave PROVEEDOR de cada registro una pareja que posea la serie de caracteres especificados, o:

```
SEARCH PRECIO>=45.50
```

Mediante el campo clave PRECIO se están buscando cifras mayores o iguales que 45.50. Algunos DBM permiten especificar todos los campos, o cualquier número de ellos, como campos clave cuando se crea el "esqueleto" del registro; otros sólo permiten otorgarles tal designación a un cierto número de campos. Como método práctico, cuantos más campos se puedan designar como campos clave, tanto mejor.

¿Me veré limitado al lenguaje que se suministre con mi ordenador, o podré programar yo mismo el DBM, prescindiendo del mismo?

Algunos DBM incluyen un lenguaje de programación incorporado que permite efectuar de forma automática sofisticadas secuencias de actividades. En un entorno de gestión, con estos lenguajes es posible escribir programas que le ahorran al operador una gran cantidad de trabajo.

Dos ejemplos notables son el *dBase II* y el *Archive*. Si sus requerimientos para base de datos tienen formas establecidas de tratar los datos, tales como "imprimir todas las ventas del día seguidas por una lista de todos los componentes en stock y una lista de todos los componentes que se encuentran por debajo del nivel de 'nuevo pedido'", entonces lo que se debe buscar es un DBM con lenguaje de programación incorporado. Si, por el contrario, no sabe cómo se utilizará la base de datos día a día, entonces bastarán las instrucciones usuales de "lenguaje de interrogación".

¿Qué he de buscar si los registros que crearé habrán de relacionarse con otros datos diferentes a los de los registros?

Los administradores de bases de datos más simples pueden trabajar con un solo archivo de registros a un tiempo. Para muchas aplicaciones esto es perfectamente adecuado. No obstante, es mucho más útil la capacidad de un DBM para trabajar con dos o más archivos al mismo tiempo. La situación clásica en la que resulta útil es la de una base de datos de control de stock en la cual cada componente puede tener más de un proveedor. Tendríamos un archivo de base de datos para "componentes" y otro archivo separado para "proveedores". Si usted efectúa una interrogación en un archivo, es sumamente útil poder efectuar interrogaciones a otro archivo, estando ambos relacionados.

Para ilustrar este punto, supongamos que llevamos una tienda de antigüedades y tenemos un archivo de base de datos "stock" sobre lo que tenemos en existencia. También podríamos llevar un registro de nuestros proveedores en el cual apuntaríamos las cosas que ellos tienen disponibles pero que aún no hemos comprado. Nuestros registros para "MOBILIARIO, SUECO" podría no tener entradas, pero una referencia cruzada a su archivo PROVEEDORES podría revelar a KURT JAKOBSEN ANTIK, GAMLA STAN 56, ESTOCOLMO. Su registro podría revelar:

DESCRIPCION:	SOFA
FABRICANTE:	ANDERSEN
FECHA:	1824
PRECIO COR. SUEC.:	12000

DESCRIPCION:	SOFA
FABRICANTE:	GRIMM
FECHA:	1874
PRECIO COR. SUEC.:	9800

e infinidad de otras entradas de artículos disponibles. En otras palabras, cuando no basta un único archivo, la referencia a uno o más archivos diferentes puede ser lo que hace falta.

Teóricamente, siempre es posible tener suficientes campos libres dentro de cualquier archivo determinado para permitir la adición de cualquier información que pudiera requerirse. En la práctica, puede que ello no sea posible. Supongamos que el señor Jakobsen nos telefonea para decirnos que hoy mismo acaba de recibir dos encantadores sofás del s. XIX que quizá podrían interesarnos. Obviamente, es mucho más sencillo entrar los detalles en el registro de JAKOBSEN que pasar por todos los campos DESCRIPCION del archivo STOCK y entrar las nuevas adquisiciones del señor Jakobsen.

Para simplificar, siempre que usted tenga una situación en la que hay más de una relación de uno a uno entre partes de un registro de base de datos con otros datos, lo que uno debe considerar fundamentalmente es un DBM de archivos múltiples.

Los campos dependientes y calculados parecen ser bastante útiles. ¿De qué forma me pueden ser de ayuda en el trabajo que realizaré con el DBM?

Algunos DBM ofrecen un refinamiento que se denomina *campos dependientes*. Éstos son campos que aparecen (durante la entrada de datos y posteriormente) sólo si son necesarios. Por ejemplo, una base de datos que posea un campo N. de HIJOS podría no visualizar campos extras si la entrada fuera 0, y NOMBRE DEL PRIMER HIJO, EDAD DEL PRIMER HIJO, etc., si la entrada fuera 1, y así sucesivamente.

Los campos dependientes son, no obstante, un mero refinamiento y muy raramente son esenciales. Algunos DBM también permiten usar los datos de ciertos campos como argumentos en operaciones aritméticas, casi como si se tratara de una minihoja electrónica. Tales sistemas permitirían, por ejemplo, multiplicar la cantidad de artículos de un campo N. EN STOCK por la cifra del campo PRECIO en cada registro, así como sumar el total de los resultados de los registros en un archivo para dar un valor total en resultado de stock. Este tipo de facilidad casi siempre viene unida a un lenguaje de programación incorporado y puede ser muy útil.

¿Cuáles son las diferencias entre los DBM activados por menú y los activados por instrucciones, y es alguno de ellos más adecuado que el otro?

Un DBM activado por menú es aquel que presenta opciones en la pantalla en cada una de las etapas de actividad. Un programa activado por instrucciones espera que usted aprenda su vocabulario de instrucciones para hacerlo funcionar (que a menudo implica una combinación tecla Control-más-letra de pulsaciones de tecla). Aunque los programas activados por menú tienden a ser más fáciles para el principiante, los activados por instrucciones son generalmente más rápidos de usar una vez que se han aprendido las instrucciones. Siempre y cuando las instrucciones se elijan de forma lógica (S para SEARCH, P para PRINT, etc.), es probable que, a la larga, el software activado por instrucciones sea más fácil de usar.

¿Qué clase de facilidades tienen incorporadas los DBM para verificar errores e impedir entradas erróneas?

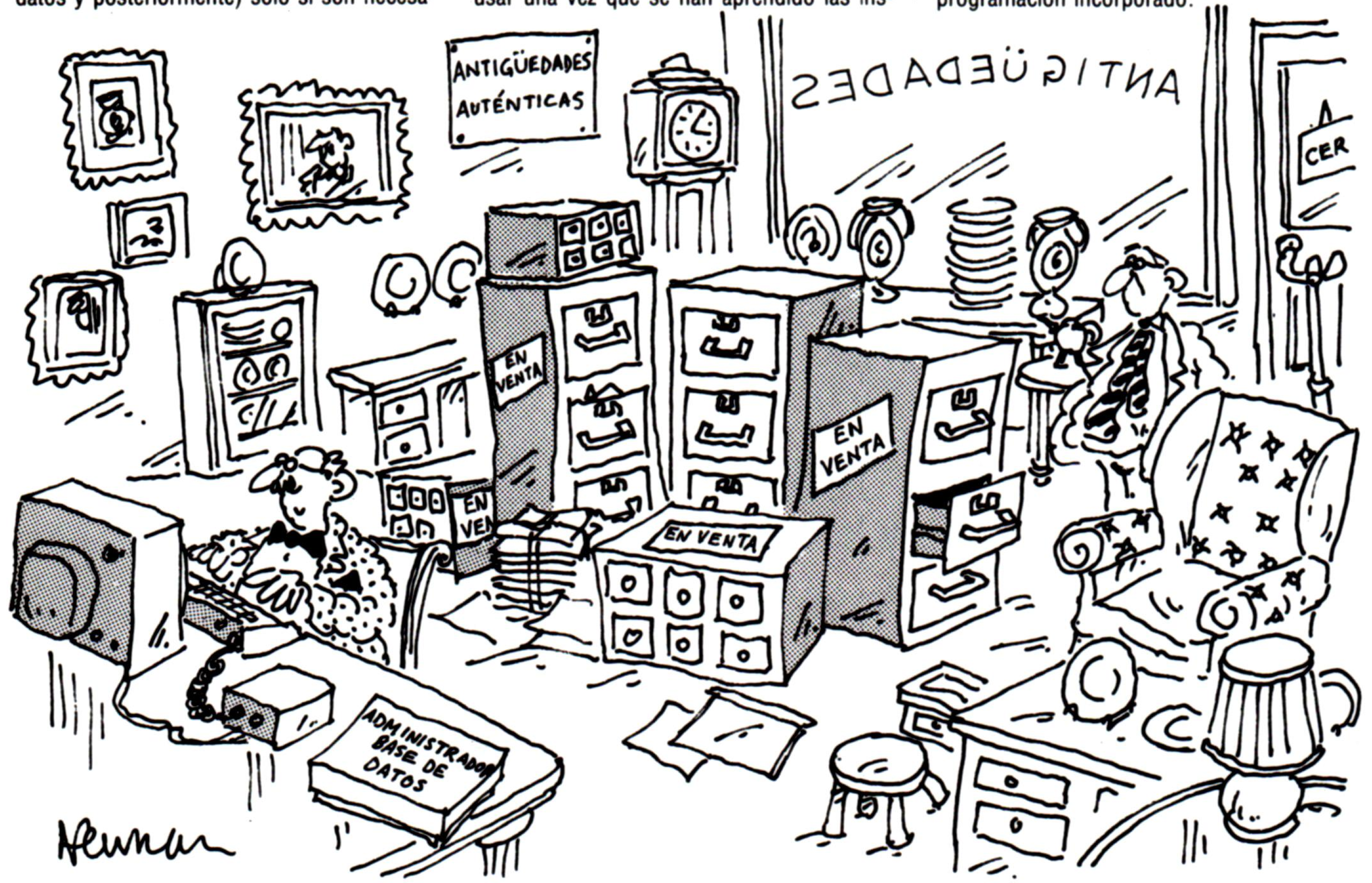
Un DBM bien diseñado le permitirá especificar qué tipo de datos se pueden entrar en cada campo, lo que contribuye, por tanto, a reducir las entradas erróneas. Por ejemplo, 31/02/85 sería rechazada automáticamente, como lo sería una entrada como:

PRECIO: reborde para montar medidor de aceite

El mejor software siempre incluye la facilidad de imponer límites sobre el tipo de datos admisible; por otra parte, en ocasiones la verificación excesivamente rígida puede ser una incomodidad. En cualquier caso, compruebe las facilidades que se ofrecen para validación de datos.

Habiendo dicho lo anterior, ¿qué DBM debo comprar?

Recomendar la mejor compra para un DBM es imposible; usted tiene que saber para qué lo utilizará antes de adquirir uno. No obstante, nuestro consejo sería, para todas las aplicaciones excepto las menos sofisticadas, un DBM basado en disco, con acceso a archivos múltiples y que, además del lenguaje de interrogación normal, poseyera un lenguaje de programación incorporado.



La curva de recorrido

En este último capítulo de nuestro proyecto revisaremos el software que hemos desarrollado hasta ahora

La rutina moverservo desarrollada como parte del software controlador del brazo reduce la torpeza inherente del brazo cuando se mueve desde una posición a otra. Los motores reciben su información angular desde una serie de posiciones de la memoria, que comienzan en ANGULO.

Para mejorar la uniformidad del movimiento del brazo, cuando los motores asumen nuevas posiciones en respuesta a cambios producidos en las posiciones ANGULO, los valores no se colocan (POKE) directamente en estas posiciones, sino en un grupo de posiciones correspondientes etiquetado NUEVAPOS. Al ser llamada, la rutina moverservo compara los valores de las posiciones correspondientes de ANGULO y NUEVAPOS e incrementa o decrementa ANGULO de la forma apropiada, de una en una unidad. Repite el proceso hasta que los valores de NUEVAPOS y ANGULO son los mismos.

Al colocar (POKE) los valores de ángulo desde BASIC en NUEVAPOS en vez de en ANGULO, efectivamente aislamos los motores contra cambios de ángulo súbitos y de gran envergadura. La tarea de desplazar un carrete de hilo de coser desde una posición conocida hasta una taza es ahora relativamente sencilla. Se puede guiar el brazo lentamente a través de los movimientos, utilizando las teclas adecuadas del teclado, y las posiciones clave de la curva se pueden guardar en una matriz. La información de la matriz se puede reproducir a cualquier velocidad insertando un factor de demora en moverservo y recorriendo paso a paso la matriz.

Es imposible introducir más mejoras. Para una primera aproximación, todos los movimientos angulares son proporcionales a los cambios de NUEVAPOS, según a qué distancia respecto al pivote se ajuste cada enlace, en comparación con el radio de la palanca activadora del motor. Por consiguiente, la rutina moverservo hace mover cada parte del brazo de forma uniforme y con una velocidad angular constante (los grados de rotación por segundo) hasta su nueva posición. Sin embargo, cada vez que alcance una nueva posición, puede ser que el brazo haya de cambiar repentinamente la dirección al iniciar su camino hacia la siguiente posición guardada en la secuencia; en otras palabras, es posible que alcance una discontinuidad.

Este problema no es tan grave como el problema original de efectuar grandes cambios súbitos de ángulo. No obstante, puede ser evitado guardando una cantidad de puntos intermedios en lugar de introducir cambios radicales entre una posición guardada y la siguiente.

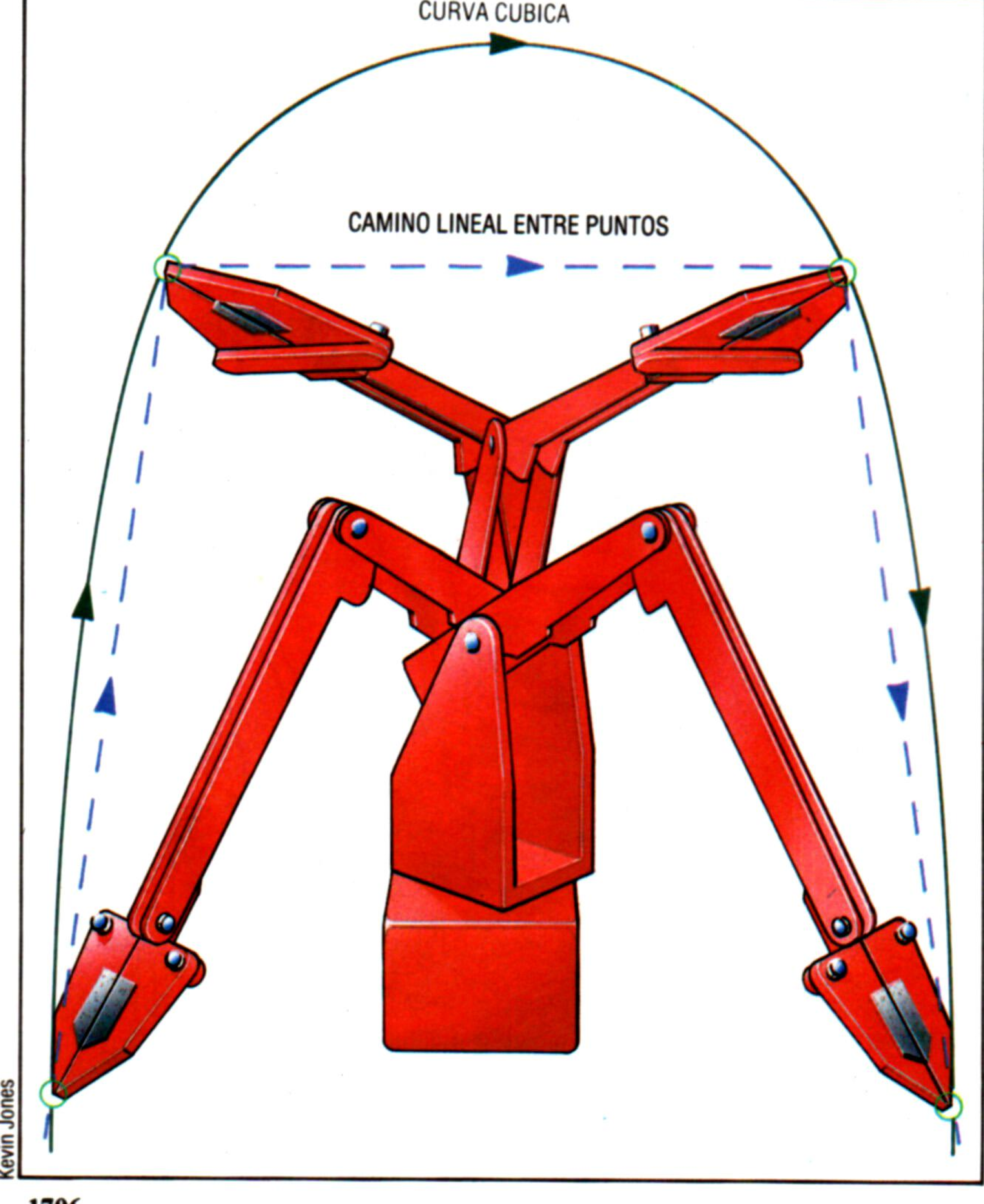
Con frecuencia se puede mejorar aún más el movimiento del brazo mediante el empleo de un software más sofisticado, pero ha de lograrse el equilibrio entre velocidad y espacio. Si el carrete de hilo se mueve especificando solamente cuatro posiciones separadas (la primera cuando se recoge el carrete, la segunda cuando se lo levanta, la tercera al asumir una nueva posición encima de la taza, y la cuarta cuando se baja el carrete a la taza) se utiliza poca memoria, pero se producen grandes discontinuidades.

Una forma de reducirlas consistiría en especificar, pongamos por caso, 60 posiciones intermedias, en cuyo caso el movimiento sería comparativamente fluido y elegante. Pero al hacer esto, el almacenamiento de los datos consumiría sustancialmente más memoria, por no mencionar el tiempo que llevaría entrar manualmente todas las posiciones.

La alternativa exige dedicar tiempo a calcular una cantidad de valores intermedios y después combinar las posiciones guardadas manualmente con las posiciones de conexión calculadas. Ahora se nos plantea el problema de cómo calcular estos valores intermedios.

Si consideramos cuatro puntos especificados, entonces se podría disponer una banda articulada de acero, para que tocara tres puntos cualesquiera de los cuatro en cualquier momento dado, en especial si los dos extremos estuvieran en posiciones convenientes. El equivalente matemático de disponer un trozo de acero articulado consiste en interpolar

Suavizando los bordes salientes
Si cuando se utiliza el programador para el brazo se guardan cuatro posiciones clave para éste, entonces, al reproducir la secuencia, el brazo se desplazará linealmente entre los puntos (indicado mediante la línea de puntos). Sin embargo, se pueden calcular varios puntos intermedios que generen una curva uniforme (conocida como *curva cúbica*) que pase a través de las cuatro posiciones guardadas. Las coordenadas de los puntos de esta curva se pueden utilizar para hacer que el brazo efectúe un barrido más uniforme a lo largo de su camino programado



Kevin Jones



puntos intermedios mediante el empleo de una *regla flexible cúbica* (así llamada porque incluye valores elevados a la tercera potencia).

El programa que ofrecemos aquí demuestra cómo se pueden interpolar los puntos intermedios que conforman una curva cúbica a partir de los cuatro puntos especificados. El movimiento resultante entre los puntos será una curva de barrido uniforme: una mejora considerable respecto a la interpolación lineal. Para usar este método con un sistema de cuatro motores, como el de nuestro brazo-robot, será necesario interpolar matemáticamente una regla flexible cúbica para cada motor.

Reglas flexibles cúbicas

BBC Micro:

```
1000 REM ********************************
1010 REM **       CURVAS CUBICAS BBC       **
1020 REM ********************************
1030 :
1040 MODE 0
1050 VDU 23,240,24,126,66,219,219,66,126,24
1060 V=7
1070 DIM X(V+3), Y(V+3), M(V+3), Z(V+3)
1080 X(2)=0: X(3)=10 :X(4)=40 :X(5)=50
1090 Y(2)=2 :Y(3)=12 :Y(4)=12 :Y(5)=2
1100 REM ampliar los extremos linealmente
1110 X(1)=X(2)+(X(3)-X(2))
1120 Y(1)=Y(2)-(Y(3)-Y(2))
1130 X(6)=X(5)+(X(5)-X(4))
1140 X(7)=X(6)+(X(6)-X(5))
1150 Y(6)=Y(5)+(Y(5)-Y(4))
1160 Y(7)=Y(6)+(Y(6)-Y(5))
1170 GLG:VDU 5:MOVE 100,700
1180 PRINT"Demostracion de una rutina"
1190 PRINT"que incorpora una curva cubica"
1200 PRINT"que une puntos"
1210 FOR I=2 TO 5
1220 MOVE X(I)*20-4,Y(I)*20+15:PRINT CHR$(240);
1230 NEXT
1240 GOSUB 1500:REM ESTABLECER AKIMA
1250 FOR X=X(2) TO X(5)
1260 GOSUB 1330:REM FUNCION AKIMA
1270 IF X=X(2) THEN MOVE X*20,Y*20
1280 IF X<>X(2) THEN DRAW X*20,Y*20
1290 NEXT
1300 END
1310 :
1320 :
1330 REM **** FUNCION AKIMA ****
1340 REM AKIMA RUCKDESCHEL LIBRO 2 (BYTE/MCGRAW-HILL)
1350 N=1
1360 REM COMPROBAR SI X SE HALLA EN EL RANGO DE LA TABLA
1370 IF X<X(1) OR X>X(V-2) THEN RETURN
1380 REM HALLAR INTERVALO DE TABLA PERTINENTE
1390 I=0
1400 I=I+1:IF X>=X(I) THEN 1400
1410 I=I-1
1420 REM COMENZAR INTERPOLACION
1430 B=X(I+1)-X(I)
1440 A=X-X(I)
1450 Y=Y(I)+Z(I)*A+(3*M(I+2)-2*Z(I)-Z(I+1))*A*A/B
1460 Y=Y+(Z(I)+z(I+1)-2*M(I+2))*A*A*A/(B*B)
1470 RETURN
1480 :
1490 :
1500 REM **** ESTABLECER AKIMA ****
1510 REM CALCULAR COEFICIENTES AKIMA
1520 FOR I=1 TO V-1
1530 REM PASAR I A I+2
1540 M(I-2)=(Y(I+1)-Y(I))/(X(I+1)-X(I))
1550 NEXT I
1560 M(V+2)=2*M(V+1)-M(V)
1570 M(V+3)=2*M(V+2)-M(V+1)
1580 M(2)=2*M(3)-M(4)
1590 M(1)=2*M(2)-M(3)
1600 FOR I=1 TO V
1610 A=ABS(M(I+3)-M(I+2))
1620 B=ABS(M(I+1)-M(I))
1630 IF A+B<>0 THEN Z(I)=(A*M(I+1)+B*M(I+2))/(A+B)
1640 IF A+B=0 THEN Z(I)=(M(I+2)+M(I+1))/2
1650 NEXT I
1660 RETURN
```

Commodore 64:

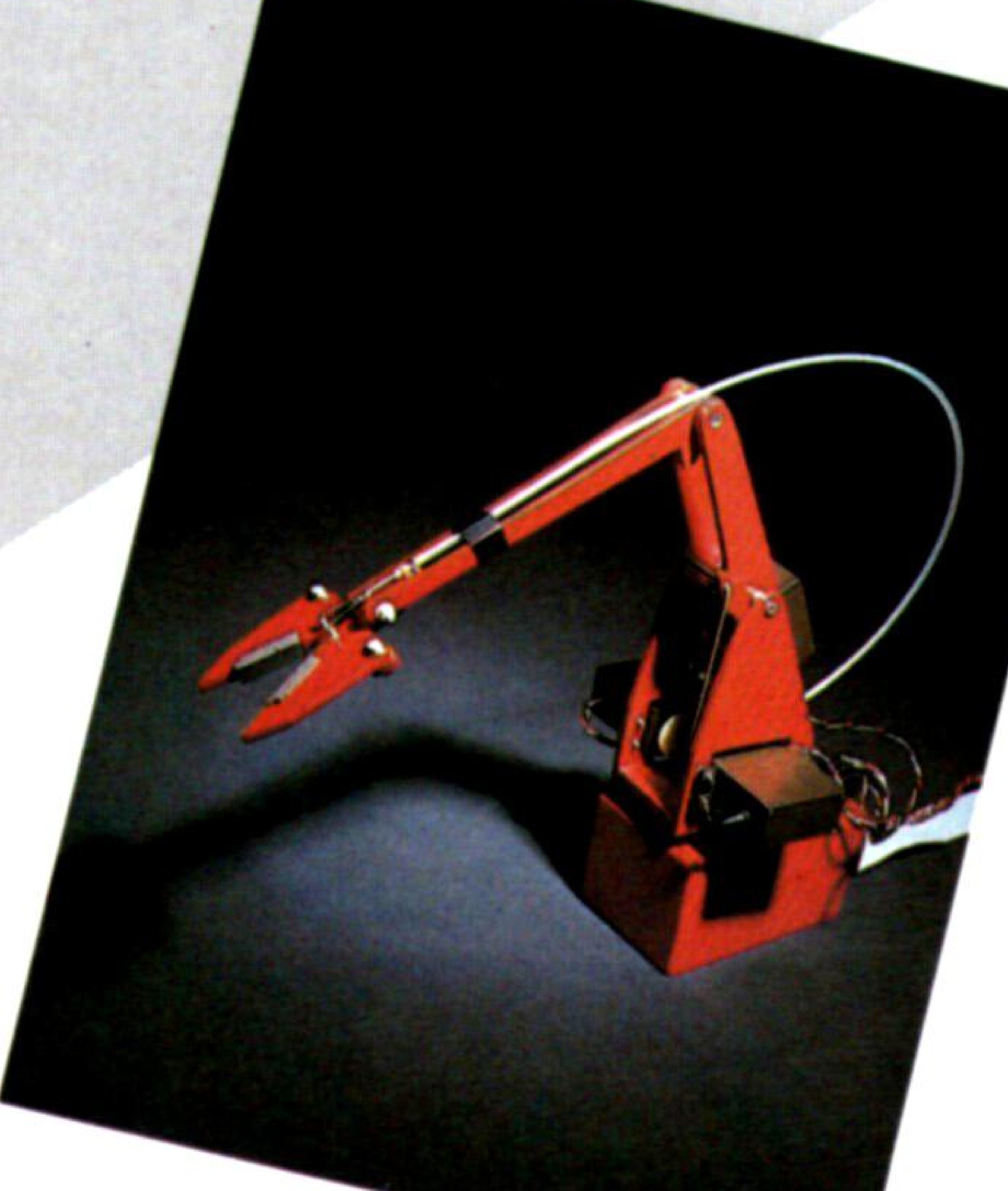
```
10 REM ****************************
20 REM **    CURVAS CUBICAS CBM    **
30 REM ****************************
40 :
50 V=7:FOR I=1 TO 25:DW$=DW$+CHR$(17):NEXT
60 DIM X(V+3),Y(V+3),M(V+3),Z(V+3)
70 X(2)=0:X(3)=5:X(4)=20:X(5)=25
80 Y(2)=2:Y(3)=12:Y(4)=12:Y(5)=2
90 REM ** EXTENDER LOS EXTREMOS LINEALMENTE **
100 X(1)=X(2)-(X(3)-X(2))
110 Y(1)=Y(2)-(Y(3)-Y(2))
120 X(6)=X(5)+(X(5)-X(4))
130 X(7)=X(6)+(X(6)-X(5))
140 Y(6)=Y(5)+(Y(5)-Y(4))
150 Y(7)=Y(6)+(Y(6)-Y(5))
160 PRINT CHR$(147)
170 PRINT"DEMOSTRACION DE UNA RUTINA QUE"
180 PRINT"INCORPORA UNA CURVA CUBICA"
190 PRINT"QUE UNE PUNTOS"
200 GOSUB 3000:REM DIBUJAR PUNTOS
230 GOSUB 1000:REM ESTABLECER AKIMA
235 J=2
240 FOR X=X(2) TO X(5)
250 GOSUB 2000:REM FUNCION AKIMA
255 IF X=X(J) THEN J=J+1:GOTO 270:REM NO DIBUJAR
260 N=X:M=Y:GOSUB 5000:PRINT CHR$(46)
270 NEXT
280 GET A$:IF A$"" THEN 280
290 END
1000 REM **** ESTABLECER AKIMA ****
1010 REM CALC COEFICIENTES AKIMA
1020 FOR I=1 TO V-1
1030 M(I+2)=(Y(I+1)-Y(I))/(X(I+1)-X(I))
1040 NEXT I
1050 M(V+2)=2*M(V+1)-M(V)
1060 M(V+3)=2*M(V+2)-M(V+1)
1070 M(2)=2*M(3)-M(4)
1080 M(1)=2*M(2)-M(3)
1090 FOR I=1 TO V
1100 A=ABS(M(I+3)-M(I+2))
1110 B=ABS(M(I+1)-M(I))
1120 IF A+B<>0 THEN Z(I)=(A*M(I+1)+B*M(I+2))/(A+B)
1130 IF A+B=0 THEN Z(I)=(M(I+2)+M(I+1))/2
1140 NEXT I
1150 RETURN
2000 REM **** FUNCION AKIMA ****
2010 N=1
2020 IF X<X(1) OR X>X(V-2) THEN RETURN:REM COMPROBAR RANGO
2030 I=0
2040 I=I+1:IF X>=X(I) THEN 2040
2050 I=I-1
2060 REM COMENZAR INTERPOLACION
2070 B=X(I+1)-X(I)
2080 A=X-X(I)
2090 Y=Y(I)+Z(I)*A+(3*M(I+2)-2*Z(I)-Z(I+1))*A*A/B
2100 Y=Y+Z(I)+Z(I+1)-2*M(I+2))*A*A*A/(B*B)
2110 RETURN
3000 REM **** IMPRIMIR PUNTOS ****
3010 FOR I=2 TO 5
3020 N=X(I):M=Y(I):GOSUB 5000:PRINT CHR$(215)
3030 NEXT I
3040 RETURN
5000 REM **** POSICION EN N,M ****
5010 PRINT CHR$(19);
5020 PRINT TAB(N);LEFT$(DW$,25-M);
5030 RETURN
```

Complementos al BASIC

Spectrum:

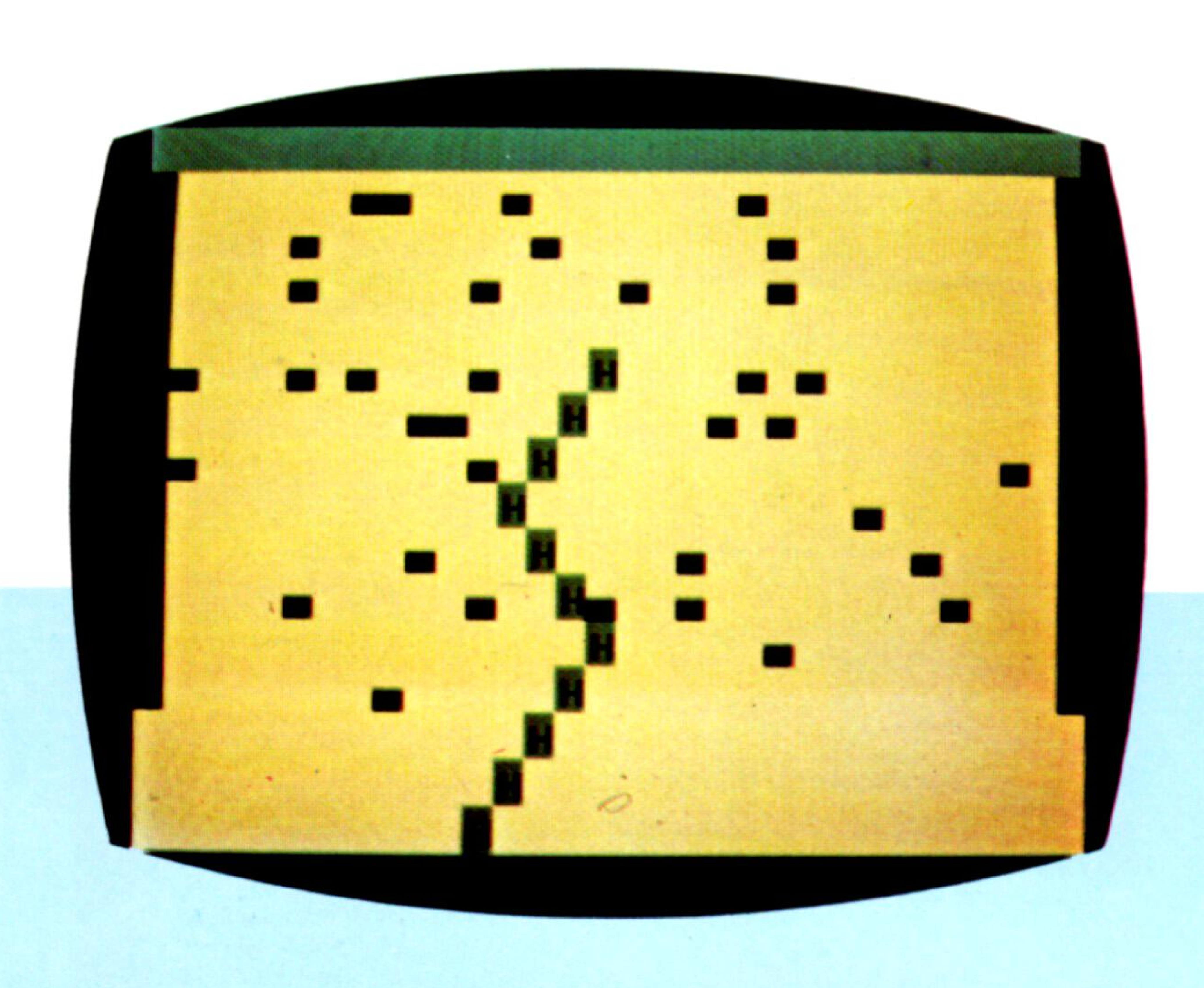
Suprimir las líneas 1040 y 1050 del listado para el BBC Micro e insertar las siguientes:

```
1170 CLS
1220 CIRCLE X(I)*5-4,Y(I)*5+15,10
1270 IF X=X(2) THEN PLOT X*5,Y*5
1280 IF X<>X(2) THEN DRAW X*5,Y*5
```



Serpiente en el C64

Este simpático juego es también un buen ejercicio de programación para crear juegos. He aquí la versión para el Commodore 64



En este juego, usted es una serpiente que se desplaza contoneándose por la pantalla. El cambio de dirección se consigue pulsando cualquier tecla. Para poder desplazarse es preciso que se alimente. Felizmente se halla rodeado por cantidad de setas. Pero, ¡cuidado! Si bien las setas azules son excelentes, ha de evitar las negras, puesto que son venenosas. Cada seta azul le proporciona las calorías necesarias para avanzar diez líneas. ¡Procure no morirse de hambre pero sin acabar envenenado!

```
5 REM ************************************
10 REM *            SERPIENTE             *
15 REM ************************************
20 GOSUB 1000
100 GET X$
110 IF X$<>"" THEN D=-D
120 T=T+D
130 IF T<L0 THEN T=L0
140 IF T>L1 THEN T=L1
150 T1=T+40
160 IF PEEK(T1)=65 THEN 500
170 IF PEEK(T1)=88 THEN S=S+10:H=H+10
180 PRINT B$;
190 P=INT(RND(TI)*40)
200 POKE 1984+P,88
210 POKE 1984+P+M,BC
220 P=INT(RND(TI)*40)
230 IF RND(TI)<0.5 THEN 260
240 POKE 1984+P,65
250 POKE 1984+P+M,RC
260 POKE T,CT
270 POKE T+M,TC
280 S=S-1
290 IF S=0 THEN 500
300 H=H+1
310 GOTO 100
500 PRINT B$;
510 POKE T,CT
520 POKE T+M,TC
530 FOR I=1 TO 500
540 NEXT I
550 IF H>RE THEN RE=H
560 PRINT CHR$(19);
580 FOR I=1 TO 10
590 PRINT B$;
600 NEXT I
610 PRINT TAB(13)"PUNTOS[1SPC]:";H;
    CHR$(19);
620 FOR I=1 TO 15
630 PRINT B$;
640 NEXT I
650 PRINT TAB(10)"PUNTUACION[1SPC]
    MAXIMA[1SPC]:";RE;CHR$(19);
660 FOR I=1 TO 20
670 PRINT B$;
690 GET X$
700 NEXT I
710 PRINT TAB(13)"OTRA[1SPC]?";
720 GET X$
730 IF X$="" THEN 720
740 IF X$<>"N" THEN 20
750 END
1000 PRINT CHR$(147)
1010 POKE 53280,5
1020 POKE 53281,13
1030 CT=19
1040 TC=9
1050 D=-1
1060 T=1404
1070 T1=T
1080 BC=6
1090 RC=0
1100 S=100
1110 M=54272
1120 B$=CHR$(17)
1130 L1=1423
1140 L0=1384
1150 H=0
1200 FOR I=0 TO 24
1210 PRINT B$;
1220 NEXT I
1230 PRINT CHR$(144);
1240 RETURN
```

Un lugar en el sol

En esta ocasión examinaremos el Wren Executive, sólido ordenador portátil construido por Thorn EMI, que cuenta con un modem y una pantalla monocromática incorporados

El ordenador Wren Executive ha tenido una historia azarosa. Construido por Thorn EMI, originalmente fue comercializado por Prism. Cuando a comienzos de 1985 la empresa quebró, pareció que el Wren iba a convertirse en otra víctima de la turbulenta industria informática. No obstante, recientemente Opus Supplies ha asumido la distribución del ordenador y tiene planes para volver a lanzar la máquina. En este capítulo analizaremos las posibilidades de que dispone el equipo para hacerse un lugar en el enormemente competitivo mercado de gestión.

El Wren Executive está diseñado para ser una máquina "portátil". Se trata, en consecuencia, de un conjunto autocontenido que consta de un ordenador, una pantalla y unidades de disco. Aunque a primera vista la carcasa parece ser de metal pintado, en realidad está construida casi por completo en plástico grabado, siendo de metal sólo la base y los contornos del teclado. En la parte posterior de la máquina hay un asa para transportarla. Al igual que muchos ordenadores portátiles, quizá el ordenador sea un poco pesado como para poder transportarlo durante mucho rato, y tal vez se describiría mejor como un ordenador de mesa que se puede transportar de un área de trabajo a otra.

El teclado posee la configuración QWERTY usual, con la única excepción de que los símbolos * y # poseen sus propias teclas. A la izquierda de las teclas de máquina de escribir hay cinco teclas de función programables que, utilizadas conjuntamente con las teclas Control o Shift, pueden cumplir hasta 15 funciones distintas. Estas teclas poseen finalidades diferentes dependiendo de la aplicación que se esté empleando, pero básicamente se utilizan para entradas de palabras clave individuales de instrucciones CP/M, tales como PIP y RENAME. En el lado opuesto del teclado se hallan las cuatro teclas del cursor y una tecla Home para llevar el cursor a la esquina superior izquierda de la pantalla. Ésta mide 150 por 105 mm y es la típica pantalla monocromática que se suministra con máquinas de gestión, con una resolución para textos de 80 por 25 caracteres. La única diferencia notoria es que el color de primer plano es naranja en lugar del verde habitual. Esto contribuye a una visualización brillante y atractiva, muy apropiada para largas sesiones de trabajo. A la derecha de la pantalla hay un par de unidades de disco flexible de 5 ¼ pulgadas.



Chris Stevens

Máquina insólita
Distribuido por Opus Supplies, el ordenador Wren Executive es una máquina portátil económica, que posee la característica insólita de ejecutar tanto BASIC BBC como el sistema operativo CP/M. Otras características incluyen unidades de disco gemelas, pantalla monocromática y un modem incorporado

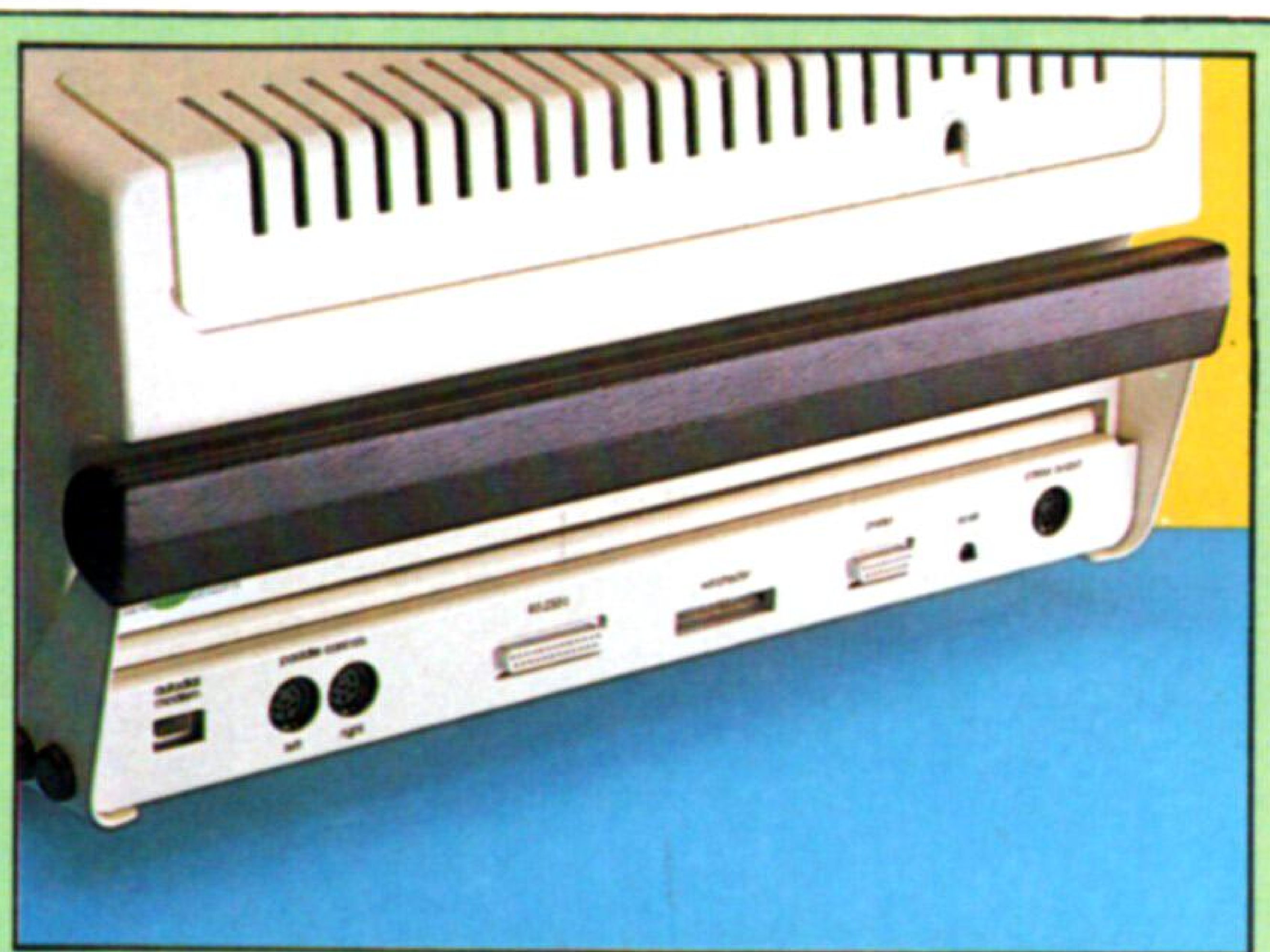
La pantalla tiene el inconveniente del diseño de la carcasa. El Wren se asienta sobre el escritorio en una posición bastante baja, sin patas para poder inclinar la máquina. La pantalla está empotrada en la carcasa hacia atrás, con un reborde de 40 mm que se proyecta hacia fuera de la pantalla. Aunque el reborde está situado en ángulo, quizá encuentre que éste oscurece la parte superior de la pantalla si usted es más bien alto o suele sentarse cerca del teclado. Ésta no es una dificultad seria, pero puede restar comodidad.

El Wren está bien provisto de interfaces. Hay varias puertas en la parte posterior que permiten conectar el ordenador a una amplia gama de periféricos. A diferencia de muchas máquinas de gestión de precio similar, el Wren posee un modem de llamada automática incorporado. El usuario del Wren está capacitado, por lo tanto, para acceder a las bases de datos más importantes, como Prestel y Micronet, sin tener que adquirir para ello ningún equipo adicional (aunque, por supuesto, usted habrá de suscribirse a una de las bases de datos antes de que se le permita establecer conexión).

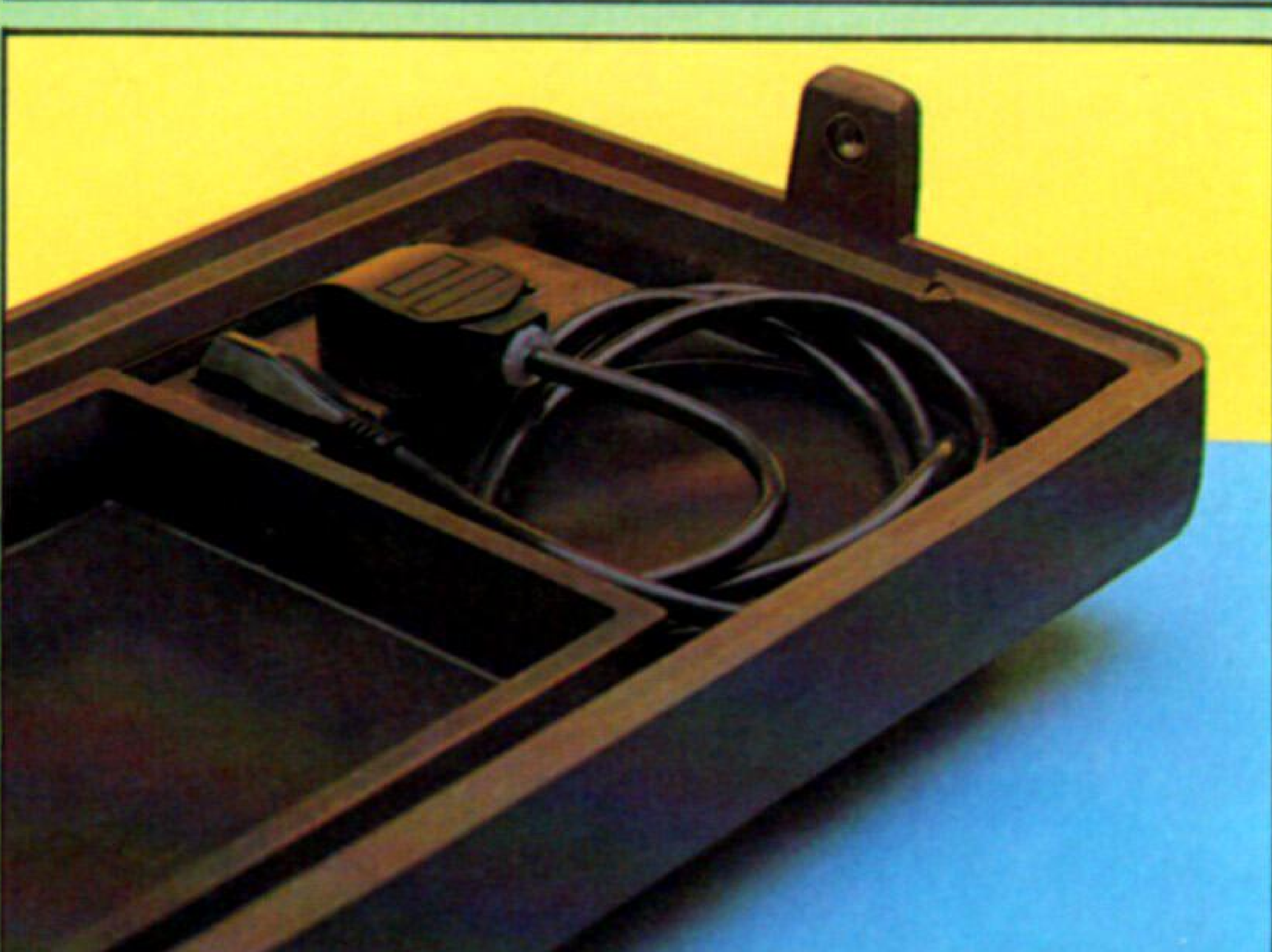
El Wren viene incluso con un cable para conectarlo a la red telefónica. El software para comunicaciones que se entrega con el ordenador permite almacenar varios números de teléfono, cuyo marcaje se realiza de forma automática cuando se desea.

Junto al conector para comunicaciones hay un par de puertas que posibilitan la instalación de dispositivos de control externos, como un ratón y palancas de mando. A continuación de estas puertas hay un conector D de 25 vías que proporciona la interface para comunicaciones seriales RS232C. Los conectores RS232 se utilizan normalmente en ordenadores de gestión para modems externos.

Abundancia de interfaces
Una de las mejores características del Wren es la riqueza de interfaces con que cuenta la máquina de forma estándar. Éstas incluyen no sólo las interfaces convencionales Centronics y RS232C, sino también un enchufe telefónico para el modem (incluyendo un cable). Asimismo, hay un conector en paralelo para facilitar la instalación de un disco rígido Winchester



Sólo hay que enchufarla
Cuando se traslada la máquina, la pantalla y las unidades de disco se protegen mediante una cubierta de plástico pesado que se coloca en la parte delantera de la máquina. Aplicando el lema de que "sólo hay que enchufarla", los fabricantes han dado el inusual paso de proporcionar con el enchufe que viene ya instalado un cable de potencia. Cuando se transporta la máquina, este cable se almacena en el interior de la carcasa



A la perfección
Al estar construido por Thorn EMI, parece natural que se haya optado por empaquetar en la máquina su propia serie de software de aplicaciones: *Perfect writer*, *Perfect filer* y *Perfect calc*. También se incluye un programa general llamado *Executive desktop*, software para comunicaciones y BASIC BBC



Obviamente, ello no es necesario en el Wren, de modo que será primordialmente usado para redes de área local e impresoras en serie.

A la izquierda de la puerta RS232 hay una interface en paralelo Winchester en la cual se puede conectar un sistema de almacenamiento en disco rígido. La adición de un sistema de disco Winchester puede añadir al sistema entre 500 Kbytes y 50 megabytes de almacenamiento adicional. Asimismo, hay una interface en paralelo compatible con Centronics que proporciona el medio para imprimir en modo *hard copy*. Por último, en el extremo derecho, hay un conector para pantalla en color RGB. Entre las interfaces para impresora y pantalla hay un diminuto botón de reset que le proporciona al sistema un arranque en frío.

El Wren está basado en el procesador de ocho bits Z80B. Éste quizá sea el punto más débil del ordenador. En una época en que los fabricantes de micros personales compiten duramente para desarrollar micros de 16 bits, el Wren parece algo anticuado en su proceso. Aunque moderadamente rápido de acuerdo a las pautas de ocho bits, no existe

Chris Stevens

Pantalla
El Wren dispone de una pantalla monocromática incorporada brillante y fácil de leer

Puerta RS232C
A través de este conector se proporciona comunicación directa con otros ordenadores y otras aplicaciones en serie

Conector para pantalla en color
Para el caso de que el usuario desee sacar partido de las facilidades de color del Wren, éste posee una puerta adicional para monitor RGB, conectable a una pantalla externa

Altavoz
Las facilidades de sonido las proporciona el propio altavoz del Wren

Interface Winchester
En esta interface en paralelo se puede enchufar un disco rígido Winchester, que aumenta notablemente la capacidad de almacenamiento externo del ordenador

Chips RAM
El Wren Executive puede direccionar hasta 400 K de RAM a través del proceso de conmutación de bancos de memoria



comparación posible con las velocidades de micros de 16 bits tales como el IBM PC o el Apple Macintosh.

Al estar basado en el procesador Z80, era natural que los diseñadores del Wren optaran por utilizar como sistema operativo el CP/M. El problema de esto es que, según los estándares recientes, el CP/M es, a pesar de su potencia, un sistema operativo más bien hostil. Los diseñadores han atenuado este problema mediante la incorporación en el sistema operativo de un procedimiento activado por menú que permite que el usuario elija una aplicación mediante un cursor y la pulsación de RETURN. El sistema solicitará entonces la colocación del disco adecuado y ejecutará automáticamente la aplicación. Por consiguiente, muchos usuarios no se verán en la necesidad de emplear el CP/M directamente, puesto que casi todo lo que necesita el usuario de gestión medio se le proporciona a través del menú.

Los discos de aplicaciones que vienen empaquetados con el ordenador son el disco de sistema CP/M, el *Perfect writer* (procesador de textos), el *Executive desktop* (paquete para llevar hora, fechas, agenda y citas), la base de datos *Perfect filer*, la hoja electrónica *Perfect calc* y un disco que contiene el software para comunicaciones y el BASIC.

El BASIC que se utiliza para programar al Wren resulta ser, sorprendentemente, el BASIC BBC estándar. Si bien la máquina posee una implementación casi completa del lenguaje, el hecho de estar basado en el procesador Z80 y no en el 6502 ha supuesto algunas diferencias, principalmente en las rutinas de entrada/salida. Asimismo, la implementación del lenguaje parece haber generado algunos errores. Por ejemplo, en las modalidades 0 y 1 el cursor desaparece de la vista. La tecla Delete no funciona en estas modalidades, y hacer retroceder el cursor e intentar reescribir provoca que el nuevo carácter se sobreimprima sobre el antiguo. Ello impide toda programación seria en estas modalidades, si bien las otras parecen funcionar perfectamente.

Expectativas inmediatas

En última instancia, el futuro éxito del Wren Executive dependerá de que pueda obtener un lugar adecuado en el mercado. Con un modem incorporado, el sistema operativo CP/M y el BASIC BBC a un precio muy competitivo, en el momento de su lanzamiento, a comienzos de 1984, el Wren parecía una oferta excepcional. Desde entonces el potencial de ventas para máquinas de gestión de ocho bits, incluso en el extremo inferior del mercado, ha venido disminuyendo rápidamente. Así y todo, la máquina sigue siendo demasiado cara como para asestar un buen golpe en el mercado del ordenador personal. Y esto es especialmente cierto ahora que Amstrad ha implementado el CP/M en su gama de micros personales.

Queda, entonces, el mercado educativo, en el cual parece ser que el Wren causará su mayor impacto. El hecho de que la máquina utilice BASIC BBC, que está muy difundido en las escuelas, junto con un modem incorporado para aprovechar las diversas bases de datos escolares que se están creando, podría hacer que la máquina les resultara atractiva a los establecimientos educativos que intentan darle a su dinero la mejor aplicación.

WREN EXECUTIVE

DIMENSIONES
440×410×250 mm

CPU
Z80B, operando a 6 MHz

PANTALLA
Pantalla para textos de 80×25, pantalla para gráficos de 640×256

INTERFACES
Puerta para pantalla, interface Centronics, puerta RS232C, enchufe telefónico, puertas gemelas para control externo, interface para disco Winchester

LENGUAJES DISPONIBLES
BASIC BBC

TECLADO
57 teclas de máquina de escribir, cinco teclas de función programables, cinco teclas para control del cursor

DOCUMENTACION
Además del manual del usuario, cada una de las aplicaciones que se proporcionan posee su propio manual. La documentación está bien organizada y ofrece al usuario explicaciones completas

VENTAJAS
La cantidad de interfaces, junto con un modem incorporado y los numerosos programas incluidos, hacen de esta máquina una excelente inversión

DESVENTAJAS
La tecnología de 8 bits empleada en el Wren se está volviendo cada vez más anticuada. Asimismo, parece haber errores en algunas de las aplicaciones

Rebelión a bordo

Existe la posibilidad de que la tripulación se amotine, se apodere del barco y emprenda el regreso, acabando, de este modo, con nuestro juego mercantil "El Nuevo Mundo"

Existen varias contingencias mayores y menores que influyen en el viaje de forma positiva o negativa. Éstas se pueden seleccionar al azar durante cada semana para su ejecución, pero no se volverán a producir tras haberse seleccionado una vez.

Son ocho los factores que pueden desencadenar un motín. A pesar de que el barco tiene capacidad para una tripulación compuesta por 16 personas, toda cantidad por encima de 12 hará que se lo considere repleto y las condiciones de hacinamiento pondrán descontenta a la tripulación. Si no se contrató un cocinero, o si éste falleció durante el viaje, la tripulación habrá de llevar a cabo tareas de cocina, lo que reducirá la calidad de la comida ¡y creará mayor conflictividad!

Dado que avistar un albatros es un buen presagio, la tripulación quedará relativamente satisfecha; pero si es abatido, lo que trae mala suerte, aumentarán las probabilidades de que se produzca el motín. Poner a la tripulación a media ración de alguna de las provisiones contribuirá aún más al descontento de la misma, y algo similar ocurrirá si las reservas de capital del barco se vuelven inferiores a los salarios adeudados. Por último, en el momento de su contratación se le aseguró a la tripulación que el viaje duraría ocho semanas. Toda dilación de este lapso provocará descontento entre los tripulantes, que se intensificará con cada semana adicional que se requiera para concluir el viaje.

Para comprobar si las condiciones son suficientes para iniciar una rebelión se crea un factor de amotinamiento, MF. Cada condición se comprueba al comienzo de una semana llamando desde el bucle principal del programa a una rutina de amotinamiento. Si el resultado de cada comprobación es positivo, se le suma un valor al factor de amotinamiento. Este procedimiento continúa hasta que MF llega a 100, en cuyo momento tendrá lugar el motín. Hay, asimismo, un factor de demora, de hasta 30, que se incluye en la evaluación semanal del factor de amotinamiento.

La línea 879 envía al programa principal a la subrutina de la línea 7200, que efectúa una comprobación semanal del factor de amotinamiento MF, que al comienzo de la rutina se establece en 0. Si en cualquier etapa del viaje se pone a la tripulación a media ración, la variable H$ se establecerá en S. Ya que obtener los alimentos suficientes es una consideración de tanta importancia para la tripulación, darles medias raciones de cualquier provisión esta-

Módulo 10: Un motín

Rutina de amotinamiento

```
879 GOSUB 7200
```

Adición al bucle principal del viaje

```
7200 REM MOTIN
7210 MF=0
7215 IF H$="S"THEN MF=MF+30
7220 NC=0
7225 FOR T=1 TO 16
7228 IF TS(T,1)=5 AND TS(T,2)<>0 AND TS(T,2)<>-999 THEN
     NC=1:T=16
7230 NEXT
7235 IF NC=0 THEN MF=MF+30
7240 IF A$="S"THEN MF=MF-20
7245 IF B$="S" THEN MF=MF+30
7250 IF CN>12 THEN MF=MF+30
7255 IF WT>M0 THEN MF=MF+30
7260 IF WK>8 THEN MF=MF+((WK-8)*10)
7275 MF=MF+INT(RND(1)*30)
7280 IF MF<75 THEN RETURN
7282 PRINT CHR$(147)
7284 IF MF>100 THEN 7300
7285 S$="LA SITUACION EN EL BARCO*":GOSUB 9100
7286 S$="ESTA EMPEORANDO*":GOSUB 9100
7287 S$="Y ALGUNOS DE LOS TRIPULANTES*":GOSUB 9100
7288 S$="HABLAN YA DE MOTIN!*":GOSUB 9100
7290 PRINT:GOSUB 9200
7292 S$=K$:GOSUB 9100
7294 GET I$:IF I$=""THEN 7294
7299 RETURN
7300 PRINT CHR$(147)
7305 PRINT:GOSUB 9200
7310 S$="LA TRIPULACION SE HA AMOTINADO*":GOSUB 9100
7312 S$="PORQUE:*":GOSUB 9100
7313 X=0
7314 IF H$<>"S" THEN 7320
7315 GOSUB 9200:X=X+1:PRINT X;
7316 S$="HAN ESTADO A MEDIA RACION*";GOSUB 9100
7318 S$="  DURANTE PARTE DEL VIAJE*":GOSUB 9100
7320 IF NC<>0 THEN 7325
7321 GOSUB 9200:X=X+1:PRINT X;
7322 S$="NO HAY COCINERO*":GOSUB 9100
7324 S$="  Y LA COMIDA ES ASQUEROSA*":GOSUB 9100
7325 IF B$<>"S" THEN 7330
7326 GOSUB 9200:X=X+1:PRINT X;
7327 S$="EL ALBATROS FUE ABATIDO!*":GOSUB 9100
7330 IF CN<13 THEN 7335
7331 GOSUB 9200:X=X+1:PRINT X;
7332 S$="LA TRIPULACION ESTA HACINADA*":GOSUB 9100
7335 IF M0>=WT THEN 7340
7336 GOSUB 9200:X=X+1:PRINT X;
7337 S$="NO HAY SUFICIENTE ORO*":GOSUB 9100
7338 S$="  PARA PAGARLES SUS SALARIOS*":GOSUB 9100
7340 IF WK<=8 THEN 7350
7341 GOSUB 9200:X=X+1:PRINT X;
7342 S$="LLEVAN NAVEGANDO*":GOSUB 9100
7343 S$="    MAS DE 8 SEMANAS*":GOSUB 9100
7350 PRINT:GOSUB 9200
7360 S$="LA TRIPULACION SE APODERA DEL BARCO*":GOSUB 9100
7362 GOSUB 9200
7363 S$="Y EMPRENDE EL REGRESO*":GOSUB 9100
7370 GOSUB 9200
7372 S$="DEJANDOTE A TI ABANDONADO*":GOSUB 9100
7373 S$="EN UN BOTE A LA DERIVA*":GOSUB 9100
7374 S$="OJALA QUE ALGUIEN TE RECOJA":GOSUB 9100
7375 PRINT:GOSUB 9200
7380 PRINT"       FIN DEL JUEGO"
7382 END
```

Complementos al BASIC

Spectrum:
Introduzca las siguientes modificaciones:

```
7282 CLS
7294 LET I$=INKEY$:IF I$=""THEN GO TO 7294
7300 CLS
```

BBC Micro:
Introduzca las siguientes modificaciones:

```
7282 CLS
7294 I$=GET$
7300 CLS
```

blecerá H$ en S para el resto del viaje. H$ se comprueba mediante la rutina de amotinamiento, sumándole 30 a MF si H$ es S. La línea 7215 comprueba H$ e incrementa MF si fuera necesario.

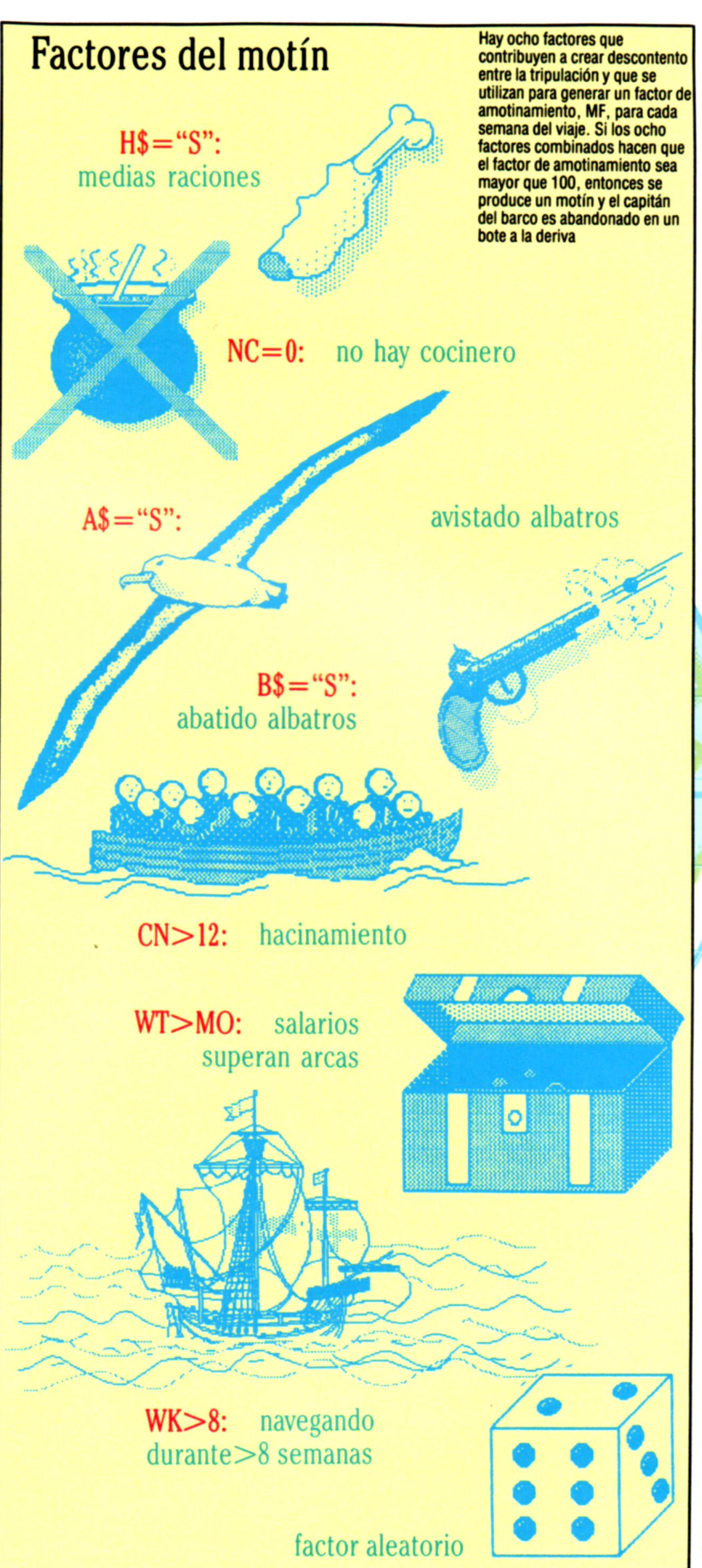
La rutina de amotinamiento comprueba entonces si hay algún cocinero a bordo estableciendo la variable NC en 0, y prepara un bucle de 1 a 16, revisando el primer elemento de la matriz de fortaleza/categoría de la tripulación TS(,), en busca de un 5, que representaría un cocinero. Si la fortaleza del cocinero no es ni 0 ni −999, lo que implicaría la presencia de un cocinero sano, NC se establece en 1. Si no se contrató ningún cocinero, o si el mismo hubiera muerto durante el viaje, NC permanece en 0. La línea 7235 comprueba NC y, si NC es igual a 0, le suma 30 al factor de amotinamiento.

Si durante la travesía ya se hubiera avistado el albatros, entonces A$ se habría establecido en S en la línea 6055 de la subrutina del albatros. Ello trae buena suerte, puesto que en la línea 7240 se le quita 20 al factor de amotinamiento. Si uno ha derribado al albatros, la línea 6162 habrá establecido B$ en S, lo que trae mala suerte, y la línea 7245 incrementa en 30 el factor de amotinamiento. La línea 7250 comprueba si el número de tripulantes es mayor que 12 y, de ser así, le suma 30 al factor de amotinamiento, haciéndose eco de este modo de los sentimientos generales de la tripulación respecto a las condiciones de hacinamiento.

La factura del total de salarios está representada por WT y el dinero que queda en las arcas del capitán mediante MO. La línea 7255 comprueba si la factura de salarios es mayor que el dinero que queda en el pozo de reserva y, de ser así, le suma otros 30 a MF. El factor de amotinamiento se incrementa en 10 cada semana adicional del viaje tras las ocho primeras semanas. La línea 7260 comprueba si la travesía ya ha durado ocho semanas y, si así fuera, la ecuación le resta 8 al número de semanas (WK), multiplica el resto por 10 y le suma este resultado a MF. En la línea 7275, se le añade a MF un factor al azar entre 0 y 29.

Si, tras comprobar todas las condiciones, el total para el factor de amotinamiento es menor que 74, la línea 7280 devuelve el control al programa principal. Si el factor es mayor que 100, la línea 7284 envía el programa a 7300, en cuyo punto se produce el motín. Si el factor se halla comprendido entre 75 y 100, las líneas 7285-7288 advierten al jugador que la situación en el barco está empeorando y que entre la tripulación se habla de amotinamiento, antes de devolver el control al programa principal.

Si la tripulación se rebela, el programa determina las causas e imprime el motivo. La línea 7314 comprueba si H$ es igual a S e informa al jugador si la tripulación ha estado a media ración durante parte del viaje. La línea 7320 comprueba si NC es igual a 0, lo que significaría que no hay ningún cocinero a bordo, y se le dirá al jugador que el motín fue consecuencia de la falta de competencia culinaria. La línea 7325 determina si el albatros fue abatido o no y, en caso afirmativo, le informa al jugador que fue un factor que contribuyó al amotinamiento. La línea 7330 comprueba si el barco estaba repleto, la 7335 determina si hay suficiente dinero para pagar a la tripulación y la 7340 examina la duración del viaje. Por último, se informará que algunos tripulantes han sido muertos y que los amotinados se han apoderado del barco y han emprendido el regreso.



Hay ocho factores que contribuyen a crear descontento entre la tripulación y que se utilizan para generar un factor de amotinamiento, MF, para cada semana del viaje. Si los ocho factores combinados hacen que el factor de amotinamiento sea mayor que 100, entonces se produce un motín y el capitán del barco es abandonado en un bote a la deriva

Ian McKinnell

Dúo dinámico

Analizaremos conceptos que nos permitirán procesar archivos de tamaño no especificado

El PASCAL ofrece dos procedimientos estándares (new y dispose) y un tipo especial de datos que, conjuntamente, proporcionan un poder y flexibilidad extraordinarios a la *asignación* y *desasignación dinámica* de memoria. Un tipo *puntero* es sencillamente una variable que, en vez de contener un valor de datos tal como un entero, "apunta a" un entero o cualquier otro objeto de datos, estructurado o no. La notación que se utiliza es similar a la empleada cuando denotamos un buffer *de siguiente dirección* de archivos (que, en efecto, apunta al siguiente registro del archivo, si bien de una forma totalmente diferente). Un tipo puntero se define colocando delante del identificador (del tipo de datos hacia el cual deseamos apuntar) una flecha hacia arriba:

```
TYPE
   MatrizLarga      = ARRAY 1..100 OF real;
   indirecta        = ↑ MatrizLarga;
```

Todos los compiladores de ISO PASCAL soportan el mismo símbolo alternativo ("@"). La definición de tipo reserva una única posición de memoria que se utilizará para retener la dirección de una matriz grande sólo cuando se la cree mediante el procedimiento new. En el ínterin, el valor del puntero permanece sin definir, tal como sucedería con cualquier otra variable, de modo que:

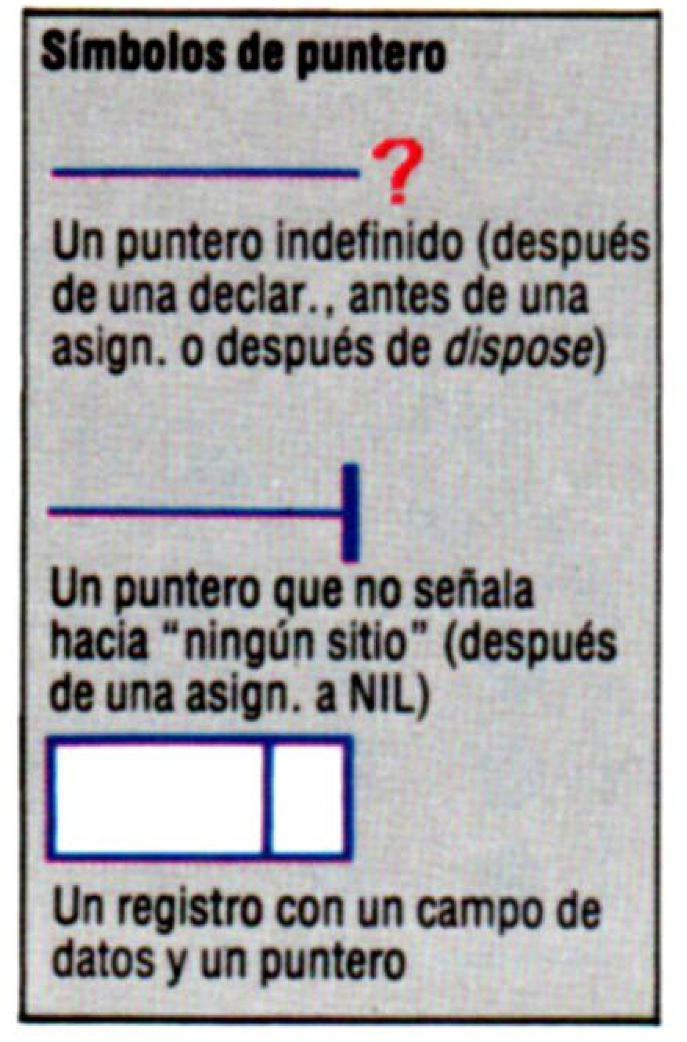
```
VAR
   direccion      : indirecta;
   numero         : integer;
```

reservaría espacio para una dirección de máquina (16 bits en un micro de ocho bits) y un entero, sin estar ninguno de ellos inicializado en ningún valor determinado.

Así como desearíamos inicializar el entero a cero antes de utilizarlo, al puntero se le podría asignar específicamente el valor especial NIL. Ésta es una palabra reservada del PASCAL, que significa tan sólo que el puntero no señala hacia ningún lugar útil, y es el equivalente del valor numérico cero (lo que significa la ausencia de cualquier número verdadero). Por consiguiente, las dos variables del ejemplo se podrían asignar así:

```
direccion    :=NIL
  numero     :=0;
```

Puesto que NIL es un valor constante que pertenece a un tipo genérico, quizá hubiera sido preferible definirlo como un identificador en el lenguaje. Resulta obvio que, en este sentido, Wirth cambió de parecer, porque en MODULA−2 la misma palabra es, en realidad, un identificador predefinido, no una palabra reservada. Para demostrar el empleo de new y dispose, por razones de simplicidad sólo utilizaremos enteros.



Cuando se necesita una variable de puntero para señalar un elemento nuevo, llamamos al procedimiento new del PASCAL, que asigna espacio para el elemento y coloca su dirección en el puntero. Después se alude al elemento de datos "desreferenciando" al puntero, p, mediante la notación p ↑ . Observe que ahora la fecha hacia arriba va detrás del identificador de puntero, al igual que en la notación buffer-archivo, y se podría imaginar como "el elemento hacia el cual apunta p". Huelga decir que es un error desreferenciar un puntero que no esté definido o que no apunte a ninguna parte.

Tras acabar con todos los datos creados mediante punteros (new), se puede llamar al procedimiento dispose del PASCAL. Éste es el opuesto de new, ya que devuelve la memoria asignada por new al "pozo dinámico" y el valor del puntero se vuelve indefinido. El programa DosMasDos lo ilustra:

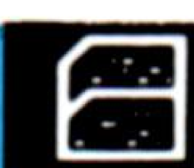
```
PROGRAM DosMasDos (output);
TYPE
   puntero          = ↑ integer;
VAR
      p1,
      p2            : puntero;
      respuesta     : integer;
BEGIN
   new(p1);
   p1 ↑ :=2;
   new(p2);
   p2 ↑ :=p1 ↑ ;
   respuesta:=p1 ↑ +p2 ↑ ;
   WriteLn (p1 ↑ ,'+',p2 ↑ ,'=',respuesta);
   dispose (p1);
   dispose (p2);
END.
```

Este método sumamente peculiar de sumar dos más dos demuestra dos puntos importantes:

- Las variables dinámicas son anónimas: no hay ningún identificador de variable (como N) que retenga el valor 2 en ningún lugar del programa; a estos elementos se alude de forma indirecta a través de los punteros.
  Tras el segundo dispose la única memoria que se utiliza es el único entero (respuesta); todos los datos dinámicos ya han dejado de existir.

Si usted ha seguido nuestra serie dedicada al lenguaje assembly, ya habrá observado que la indirección implícita en el uso de punteros es análoga al direccionamiento indirecto a nivel de lenguaje máquina. Sin embargo, hay grandes diferencias en el uso de la indirección en un lenguaje de tan alto nivel como el PASCAL. En primer lugar, nunca sabemos (o no necesitamos saber) cuáles son las verdaderas direcciones. La única "dirección absoluta" de que dispone el programador de PASCAL es NIL.

- Asimismo, somos libres de utilizar, reclamar y posteriormente reutilizar la memoria disponible sin necesidad de organizar nosotros mismos ninguna "recolección de basura". Es el PASCAL el que se ocupa de la gestión de la memoria, y la única información que quizá podríamos necesitar es cuánta



memoria queda. Esto se obtiene mediante la función no estandarizada MemAvail o, en algunas implementaciones, como en el ISO PASCAL Acorn, mediante la función Free. Ésta devuelve la cantidad de bytes de memoria que quedan disponibles.

Otra ampliación útil es la función SizeOf (IdentificadorTipo), que devuelve el tamaño, en bytes, de cualquier tipo determinado. La RAM para el usuario se divide en dos estructuras internas: la pila y el *heap* (montón).

La pila se emplea para llamadas a procedimientos y funciones, almacenando sus direcciones de retorno, datos locales y valores devueltos. Todos los datos dinámicos se asignan al montón. Éste opera de forma similar a la pila, a excepción de que no es una estructura de datos LIFO (último en entrar, primero en salir) y el montón comienza en el extremo opuesto de la memoria para el usuario, creciendo en dirección a la pila. El uso excesivamente ambicioso de new y/o la recursión podría producir una "colisión pila-montón", pero ésta se puede evitar mediante:

```
IF SizeOf (cosa)>PerCent*MemAvail THEN..
```

donde cosa es el identificador de tipo de los datos que están por crear y PerCent es un valor del orden de 0.7, permitiendo el 30 % de la memoria para la pila y un 70 % para el montón. Si, como suele ser el caso, MemAvail/Free opera como una "línea de marea alta", se podría mantener un "contador de basura" adicional de elementos desechados.

Estructuras enlazadas

El verdadero poder de los punteros queda de manifiesto cuando creamos estructuras enlazadas tales como árboles, listas de enlace simple o doble, estructuras circulares, etc. Si consideramos el problema de series de caracteres, podemos usar, y a menudo lo hacemos, una matriz de algún tamaño estipulado, pongamos 80 caracteres. Si tenemos una matriz de series para almacenar un documento, por ejemplo, cada línea en blanco consumirá aún 80 bytes de almacenamiento. Igualmente, no podemos representar líneas de longitud superior a 80 caracteres: toda la estructura de datos es, sencillamente, demasiado rígida.

Un compilador de PASCAL ha de distinguir identificadores de cualquier longitud. Entonces, ¿cómo podemos, por ejemplo, reproducir con precisión esta característica del mundo real de tamaño variable? La estructura natural a utilizar sería un registro que contuviera dos campos: uno para cada elemento de datos (chars en este caso) y otro de puntero que señalara hacia el registro siguiente, si lo hubiera, de la lista.

```
TYPE
      serie        = ↑ caracter;
      caracter     =RECORD
                       ch :char;
                       siguiente :serie
                       END;
VAR
      linea      :serie;
BEGIN
   linea:=NIL;
            etc.
```

Listas enlazadas

VAR
linea :serie; ?

El PASCAL reserva espacio para una dirección (indefinida) de la memoria que sigue a una definición de tipo

linea :=NIL;

Un puntero se puede inicializar a "cero" asignándole el valor especial NIL

new(linea); ? ?

El procedimiento *new* asigna espacio para datos en la memoria y almacena la dirección de la zona reservada en la variable de puntero

linea ch siguiente

linea ↑ .ch:='a'; 'a' ?

A los elementos de datos no se alude de forma explícita, sino que se los "direcciona indirectamente" mediante la notación ↑

linea ↑ .siguiente:=NIL; 'a'

P :=linea;

```
WITH P ↑ DO
  BEGIN
    new(siguiente);
    siguiente ↑ .chr:='b'
  END
```

'a' 'b' ?

Una lista de enlace simple

Liz Dixon

En sucesión
El PASCAL ofrece al programador la posibilidad de implementar "listas enlazadas", poderosas estructuras de datos en las que cada elemento apunta al siguiente elemento de la lista. Las listas pueden ser de enlace simple (entre elementos sucesivos), de enlace doble (cada elemento posee punteros tanto hacia el elemento siguiente como hacia el elemento anterior de la lista), o circulares (donde el último elemento de una lista de enlace simple señala hacia atrás, hacia el primer elemento de la lista)

Una serie vacía se representa asignándole a la serie el valor NIL. Cualquier otra secuencia de caracteres exigirá un nuevo registro que contenga cada char y otro puntero que apunte al siguiente registro. El último registro tendrá su campo siguiente inicializado en NIL, de modo que se pueda detectar el final de la serie. Un procedimiento para imprimir la serie sería, entonces:

```
WHILE linea<>NIL DO
   BEGIN
     write (linea ↑ .ch);
     linea:=linea ↑ .siguiente
   END
```

Observe que la estructura de datos es recursiva, puesto que se define en términos de sí misma. Al campo de puntero no se le puede dar un tipo que se haya definido totalmente, de modo que se permite una referencia "hacia adelante"

Programa de lista circular

```
PROGRAM   ListaCirc   (input, output);
(______________________________________________
___
___ Intención: Insertar registros que contienen
___ datos mezclados en una lista circular
___ asignada dinamicamente. Los datos se colocan POR orden
___ DE un campo alfabeticò clave (Nombre) y, por consiguiente,
___ son innecesarios algoritmos de clasificacion.
______________________________________________

CONST
        LongitudSerie      =25;
        espacio            =' ';

TYPE
        Cardinal           = 0..MaxInt;
        TamañoSerie        = 1..LongitudSerie;
        serie              = PACKED ARRAY [TamañoSerie]
                                   OF char;
        cosa               = RECORD
                                  Nombre        :serie;
                                  (..otros campos)
                                  deuda         :Cardinal
                               END; (cosa)
        puntero            = ↑ nudo;
```

La línea circular
Este programa le permite insertar registros que contengan datos mezclados en una lista circular asignada dinámicamente. Los datos se colocan por orden de un campo clave alfabético (Nombre) y, por consiguiente, no se requieren algoritmos de clasificación

```
nudo = RECORD
          item : cosa;
          siguiente : puntero
       END; {nudo}
VAR
  lista : puntero;

{11111111111111111111111111111111111111}
PROCEDURE LeerCosas ( encabezamiento: puntero);
 VAR
   datos : cosa;
   IK : boolean;
 {2222222222222222222222222222222}

PROCEDURE SaltoarBlancos ( VAR F : text );
  {ignorar espacio en blanco de adelante}
 VAR
   hecho : boolean;
 BEGIN
  hecho := EoF (F);

  WHILE NOT hecho DO
   BEGIN
    hecho := (F ↑ >espacio) OR EoLn (F);

    IF NOT hecho THEN
     BEGIN
      get (F);
      hecho := EoF (F)
     END
 END; {SaltarBlancos}
 {2222222222222222222222222222222}
PROCEDURE LeerFicha ( VAR F : text;
                      VAR N : Cardinal;
                      VAR OK : boolean);
 {recoger un numero no negativo}
TYPE
  simple = 0..9;
VAR
  digitos : SET OF char;
  hecho : boolean;

  {3333333333333333333333}
FUNCTION Valor (digito :char) : simple;
 {convertir char A valor numerico}
 BEGIN
  Valor := ord (digito) – ord ('0')
 END; {Valor}
   {33333333333333333333333}
FUNCTION Legal ( N : Cardinal;
                 digito : char)
  {estamos POR superar MaxInt ?} : boolean;
 BEGIN
  IF N = MaxInt DIV 10
   THEN
    Legal := Valor (digito)<=MaxInt MOD 10
   ELSE
    Legal := N<MaxInt DIV 10
 END; {Legal}
   {33333333333332222222222222}
BEGIN {LeerFicha}
 digitos := ['0' ..'9'];
 OK := NOT EoF (F);

 IF OK THEN ({hallado un digito?]
  OK := F ↑ IN digitos;

 N := 0;
 hecho := NOT OK;

 WHILE OK AND NOT hecho DO
  BEGIN {llevar N A base 10:}
   N := 10*N+Valor (F ↑ );
   get (F);
   hecho := EoF (F);

   IF NOT hecho THEN
    hecho := NOT (F ↑ IN digitos);

   IF NOT hecho THEN
    OK := Legal (N,F ↑ )
  END;

 IF NOT EoF (F) THEN
  ReadLn (F)

END; {LeerFicha}
{2222222222222222222222222222222}
PROCEDURE LeerLinea ( VAR S : serie );
  {leer una linea DE chards desde el teclado}
VAR
  indice : 0..LongitudSerie;
  simbolo : chard;
BEGIN
 SaltarBlancos (input);
 indice :=0;

 WHILE NOT EoLn AND (indice<Longitud Serie) DO
  BEGIN {colocar un char en la serie}
   indice := succ (indice);
   read (simbolo);
   S [indice]:=simbolo
  END;

 IF indice<LongitudSerie
  THEN {rellenar con NULos}
   FOR indice := indice+1 TO LongitudSerie DO
    S [indice] := chr (0)
  ELSE
   IF NOT EoLn THEN {demasiados chars}
    WriteLn ('ATENCION–truncada la entrada');

 ReadLn

END; {LeerLinea}
 {2222222222222222222222222222222}
PROCEDURE Insertar ( lista : puntero;
                     datos : cosa);
VAR
  END

  explorador,
  enlace : puntero;
  alpha : serie;
BEGIN {colocar datos en el nudo de cabeza:}
 lista ↑ item := datos;
 explorador := lista;
 alpha := explorador ↑ .siguiente.item.Nombre;

 WHILE datos .Nombre<alpha DO
  BEGIN {recorrer la lista}
   explorador := explorador ↑ .siguiente;
   alpha := explorador ↑ .siguiente ↑ .item.Nombre
  END;
   {ninguna comprobación DE duplicados}
 new (enlace);            {crear otro nudo}
 enlace ↑ .item := datos; {insertar los datos,}
 enlace ↑ .siguiente:=explorador ↑ .siguiente; {& enlace}
 explorador ↑ .siguiente:=enlace      {en la lista}

END; {Insertar}
 {22222222222222222111111111111111}

BEGIN {LeerCosas}

write ('Nombre?' );
 LeerLinea (datos.Nombre);
  {detenerse en una entrada nula:}
 WHILE datos . Nombre [1] <> chr (0) DO
  BEGIN
   REPEAT
    REPEAT
     write ('Cantidad adeudada ?':20);

    IF EoLn (input) THEN
     ReadLn (input);

    SaltarBlancos (input)

   UNTIL NOT EoLn (input);

   LeerFicha (input, datos. deuda. OK);

   IF NOT OK THEN [ningun digito]
    WriteLn ('***ERROR–':20,
     'por favor vuelva a entrar***')

  UNTIL OK;

  Insertar (encabezamiento, datos);
  WriteLn ('(RETURN al acabar)' :40);
  write ('Nombre?');
  LeerLinea (datos. Nombre)
 END

END; {LeerCosas}
{11111111111111111111111111111111111111}
PROCEDURE Visualizar (lista :puntero);
 VAR
   imprimir : puntero;
 BEGIN
   {primero saltar el nudo de cabeza ficticio:}
  imprimir := lista ↑ .siguiente;
  page (output);
  WriteLn ('Lista ordenada:');
  WriteLn;
   {algun dato (mas) A imprimir?}
  WHILE NOT (imprimir=lista) DO
   BEGIN
    WITH imprimir ↑ .item DO
     WriteLn (deuda : 8, Nombre : 30);

    WriteLn;
    imprimir := imprimir ↑ .siguiente
   END

END; {Visualizacion}
{11111111111111111111111111111111111111}

BEGIN {ListaCirc–PROGRAMA principal}

 new (lista); {crear un nudo de cabeza ficticio}
 lista ↑ .siguiente := lista; {hacer que se apunte A si mismo}
 page (output);
 WriteLn ('Entrar datos : primero el apellido, luego');
 WriteLn ('la cantidad adeudada en ptas. (sin decimales) :');
 WriteLn;
 Leer Cosas (lista);      {poner datos en la lista}
 Visualizar (lista);      {ahora imprimirla POR orden}
 WriteLn ('---hecho---' : 40)

END.
```



Tecnomúsica

El SID (Sound Interface Device: dispositivo interface para sonido) es el responsable de las posibilidades sonoras del Commodore 64

El sonido emitido por el Commodore 64 suele canalizarse a través del enchufe RF que conecta directamente con el televisor. Pero la salida sonora puede igualmente ser dirigida a través de un enchufe de audio/video, a un sistema *hi-fi* para su reproducción o grabación perfecta. En este capítulo estudiaremos los principios subyacentes en el software que va a convertir al Commodore 64 en una caja de ritmos.

La calidad del resultado final se mejora sensiblemente si la reproducción se realiza por medio de un sistema de alta fidelidad.

Además de crear sofisticados sonidos bajo control de software, el chip SID puede aceptar señales acústicas externas. Tales señales pueden ser generadas por hardware externo electrónico (que quizá incorpore otro chip SID) o por instrumentos musicales tales como la guitarra eléctrica. Esta señal externa puede mezclarse con la salida acústica del chip SID y ser procesada por sus filtros. Sin embargo, debemos aconsejar cautela en el empleo de las interfaces de E/S, ya que una conexión incorrecta de las líneas externas puede dañar seriamente al ordenador.

Le sugerimos que consulte el manual del fabricante sobre las salidas correctas y los niveles antes de realizar ninguna conexión externa.

Para comenzar el análisis, debemos enfocar nuestra atención en cómo se obtiene un sonido musical (periódico) a partir de notas puras individuales. Consideraremos después otro método alternativo para conseguir el mismo resultado. Por medio del control de la envoltura de un determinado sonido periódico, podemos introducir armónicos en diversas proporciones. Esto en la práctica significa que podemos crear casi cualquier sonido periódico que deseemos mediante una sencilla alteración de unos pocos "ingredientes" de la envoltura de un diapasón. Estos ingredientes —o puntos de control— son conocidos por las siglas inglesas ADSR (*attack, decay, sustain, release*: subida, bajada, retención, final).

Por desgracia el BASIC del Commodore 64 no proporciona muchas facilidades para el tratamiento de los sonidos. La programación sonora se basa fundamentalmente en las instrucciones PEEK y POKE. Dando a la variable SID el valor 54272 (la dirección de base del chip SID) hacemos que las direcciones SID y siguientes hasta la SID+28 controlen el chip del sonido y, por tanto, toda la capacidad sonora del Commodore 64.

Por último, proporcionaremos un programa en código máquina que creará una máquina tambor o de ritmos gestionada por interrupciones y a tres voces. Esta cuña en código máquina puede ser controlada desde el BASIC, pero las percusiones continuarán con independencia del programa en BASIC. Lo que significa que, con ligeras modificaciones, podemos proporcionar sonidos de fondo a cualquier programa en BASIC.

El sonido llega a nuestros oídos en forma de vibraciones periódicas del aire. El número de vibraciones por segundo se denomina *altura* (o *frecuencia*) de un sonido. El umbral inferior de sensibilidad del oído humano está cifrado en 15 ciclos por segundo (15 hertzios). Una nota pura de 100 Hz nos parece baja. La nota *la* inmediatamente superior al *do* normal o central tiene 440 Hz como cifra universalmente aceptada. Si se dobla la frecuencia de una nota, elevamos su altura a una octava justa. Un oído humano normal puede percibir hasta 10,5 octavas. Los osciladores de tres notas del chip SID sólo se mueven en un arco de ocho octavas (aproximadamente desde 0 Hz hasta 4 000 Hz).

Jean Fourier (1768-1830), físico francés, fue el primero que observó que toda forma de onda periódica puede descomponerse en una nota pura fundamental mezclada con otras notas cuyas frecuencias son múltiplos de dicha nota fundamental. Son los conocidos *armónicos*. El timbre especial que distingue una nota de otra de igual frecuencia es precisamente obra de los armónicos que la acompañan.

Una onda senoidal pura, que corresponderá a una nota pura, es en esencia una señal analógica, lo que quiere decir que no es tan fácil de producir en un dispositivo digital con sólo dos niveles de voltaje, 0 V o 5 V. Por ello, en lugar de generar frecuencias puras y mezclarlas después para obtener el sonido periódico deseado, debemos adoptar otro método con un ordenador personal.

El chip SID está equipado con tres voces, cada una de las cuales puede generar una de las cuatro ondas periódicas distintas, que son:

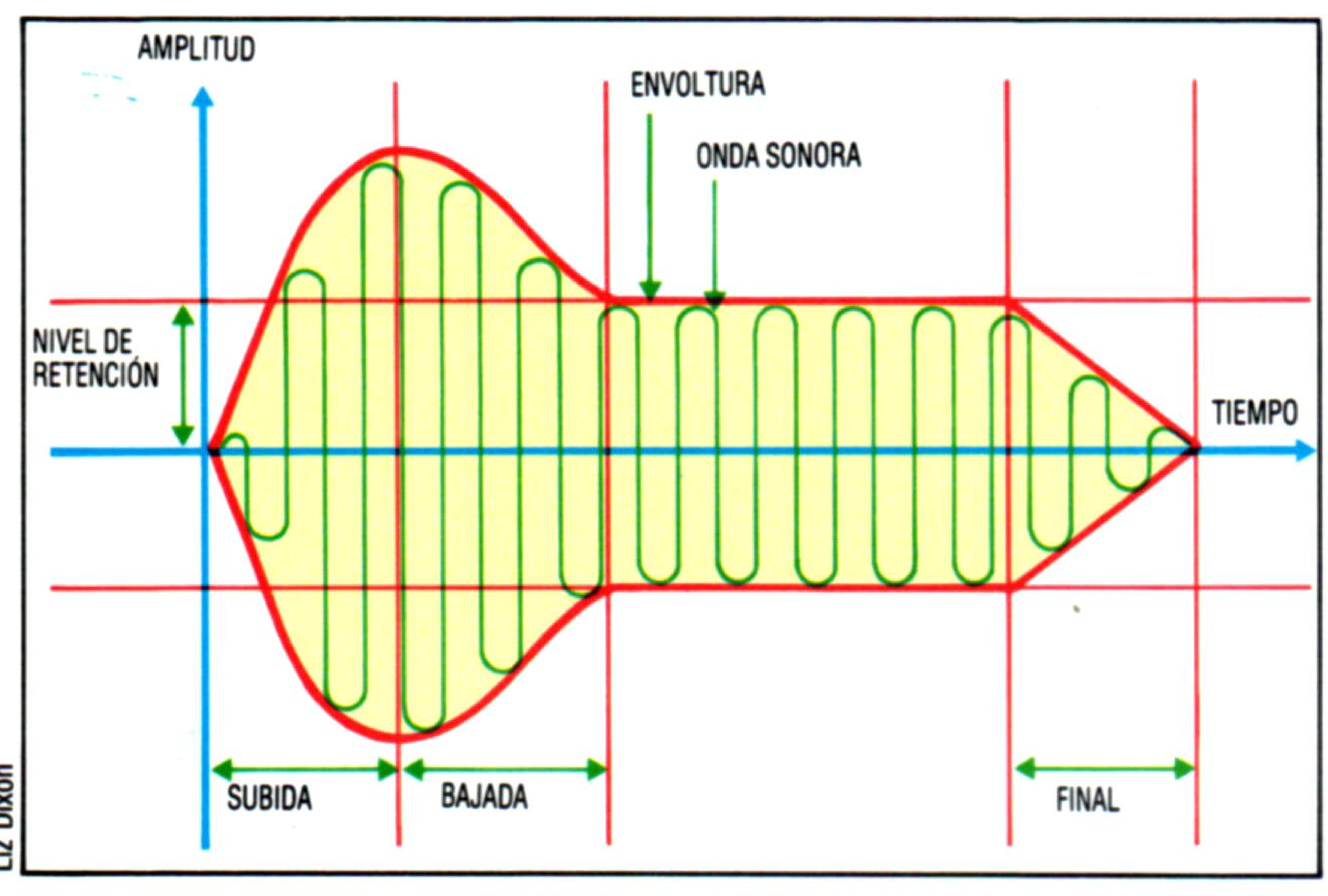
1) *Onda sierra:* contiene todos los armónicos. El armónico enésimo tiene una intensidad proporcional a 1/N.
2) *Onda triangular:* sólo contiene los armónicos impares. Para un N impar, el armónico enésimo tiene una intensidad proporcional a $1/N^2$.
3) *Onda rectangular:* es una onda cuadrada que tiene armónicos pares proporcionales a 1/N. Cambiando la "amplitud de pulsación" se puede obtener una extensa gama de ondas rectangulares, cada una de las cuales con su propia mezcla de armónicos.
4) *Ruido blanco:* es una mezcla aleatoria de frecuencias que se emplea sobre todo para efectos especiales.

El contenido armónico de un sonido también puede alterarse filtrándolo. El chip SID presenta tres tipos de filtro: paso inferior, paso de banda y paso superior. Así, un filtro de paso inferior dará paso a todas las frecuencias que estén por debajo de un valor determinado y atenuará todas las fre-

cuencias por encima de éste. Con estas facilidades y el control de la envoltura ADSR pueden producirse casi todos los sonidos.

Los programas generadores de habla humana pueden realizarse de varias maneras. Aquí describiremos un método que promete una calidad razonable y un vocabulario ilimitado. En este método, las unidades fundamentales del habla —los fonemas— se codifican en una tabla de valores ADSR. En inglés (para cuyo idioma estamos pensando este método) se conocen 52 fonemas diferentes; por ello, su codificación no presenta mayores problemas. Posteriormente emplearemos un programa en código máquina para traducir el texto en ASC (o sea, el ASCII del Commodore) a una corriente de códigos de fonemas, que se enviará después al chip SID empleando la tabla ADSR. Esto no resulta tan fácil como se explica, puesto que las reglas de traducción de un texto en fonemas son muy complejas. La calidad de los sonidos finales dependerá de esta parte del programa precisamente. El esquema es perfectamente aplicable al Commodore 64 y se comercializan diferentes productos que utilizan esta técnica.

Control de la envoltura



Liz Dixon

Envolturas finísimas
La calidad de una nota (el conjunto de características que nos permiten discernir entre una nota de piano y otra de violín, p. ej.) depende del diseño de la envoltura. En la sintetización electrónica de los sonidos la envoltura se considera integrada por cuatro fases distintas. Son conocidas por subida, bajada, retención y final o por las siglas inglesas ADSR. La longitud de cada fase de la envoltura ADSR puede ser definida con valores colocados (POKE) en los registros del SID. Esto es lo que nos permite sintetizar los sonidos de diferentes instrumentos en el Commodore 64

El dibujo de ADSR muestra la forma general de una nota musical subrayando aquellos aspectos que el chip SID puede controlar. Los cuatro factores de la ADSR son:

1) *Subida (attack):* tiempo de elevación de una nota.
2) *Bajada (decay):* tiempo de bajada de una nota a un nivel estable.
3) *Retención (sustain):* volumen de un nivel estable.
4) *Final (release):* tiempo empleado en bajar hasta cero el volumen de la nota.

Las tres primeras zonas de una nota se controlan una a una por cuartetos o *nybbles* (cuatro bits) en los registros del SID. Lo que significa que cada uno de estos parámetros toma valores entre 0 y 15. La relación entre los valores que han de ser colocados (POKE) en los registros del SID y la temporización real está ilustrada en el siguiente cuadro:

Valor	Subida (tiempo/ciclo)	Relación bajada/final (tiempo/ciclo)
0	2 ms	6 ms
1	8 ms	24 ms
2	16 ms	48 ms
3	24 ms	72 ms
4	38 ms	114 ms
5	56 ms	168 ms
6	68 ms	204 ms
7	80 ms	240 ms
8	100 ms	300 ms
9	250 ms	750 ms
10	500 ms	1.5 seg
11	800 ms	2.4 seg
12	1 seg	3 seg
13	3 seg	9 seg
14	5 seg	15 seg
15	8 seg	24 seg

La longitud de la zona de retención se calcula mediante un bucle de retardo. Con el cuadro anterior, los valores ADSR para un sonido de violín serán:

ADSR	Tiempo	Valor en POKE
A	500 ms	10
D	300 ms	8
S	—	—
R	750 ms	9

La obtención de un sonido en el Commodore 64 ha de incluir como mínimo los siguientes pasos. Así:

- Paso 1: dar volumen mediante:

```
POKE SID+24,15
```

- Paso 2: seleccionar la ADSR. Por ejemplo:

```
POKE SID+5,9 :REM VOZ#1, SUBIDA/BAJADA
POKE SID+6,0 :REM VOZ#1, RETENCION/FINAL
```

- Paso 3: seleccionar la frecuencia de cada oscilador. Por ejemplo:

```
POKE SID+1,25 : REM VOZ#1,BYTE SUP. DE FRECUENCIA
POKE SID,0 : REM VOZ#1,BYTE INF. DE FRECUENCIA
```

- Paso 4: seleccionar el tipo deseado de forma de onda. Por ejemplo:

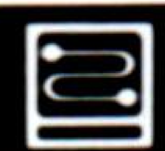
```
POKE SID+4,33 : REM VOZ#1,ONDA SIERRA
```

En este paso se empieza a emitir el sonido (lo que se denomina abrir la "puerta").

- Paso 5: bucle de retardo mientras se emite el sonido en el nivel de retención.
- Paso 6: finalizar la forma de onda. Por ejemplo:

```
POKE SID+4,32 : REM FINAL ONDA SIERRA
```

El procedimiento más sencillo de programar una melodía en un Commodore 64 consiste en establecer los valores iniciales de la ADSR y construir un bucle FOR...NEXT que lea (READ) los datos (DATA) de los bytes *hi/lo* de la frecuencia. Si en DATA incluimos ceros, podemos variar el ritmo de las diferentes voces sin cambiar el retardo.



Caja de ritmos

El programa emplea una cuña IRQ, de modo que, una vez establecidas, las percusiones seguirán sonando con independencia de lo que se esté realizando en BASIC. Tiene además otro aspecto interesante en el modo como comprueba desde el código máquina cuál ha sido la tecla pulsada, inspeccionando el contenido de la posición 197 ($00C5). Se trata de una posición de la página cero que contiene un valor que indica la última tecla pulsada. Un programa en código máquina le puede evitar el problema de una llamada núcleo para obtener la pulsación de una tecla, siempre que $00C5 sea empleado con la suficiente frecuencia.

Nótese que a causa del driver del BASIC, que no deja libre el buffer del teclado por medio de GET o INPUT, el programa debe poner a cero el puntero contenido en la posición 198 ($00C6) en el momento de la salida. Esta posición es de la página cero, que contiene generalmente el número de teclas pulsadas almacenado en el buffer de teclado.

Al utilizar el programa, cada tambor dispone de 16 "trozos de tiempo" en los que puede sonar, y se visualizan en una cuadrícula en pantalla. La elección del 16 facilita la obtención del ritmo rock en cuatro por cuatro. El cursor de sprites se mueve a una de las celdillas de la cuadrícula y en el momento de situarse en la posición correcta se activa/desactiva un "trozo de tiempo" con la tecla Return obteniéndose el ritmo mediante la tecla f1. El menú sobre la pantalla indica las opciones disponibles. Dado que los sonidos se controlan por interrupciones, se puede editar en pantalla el modelo de ritmo mientras éste se ejecuta.

El programa en BASIC incluye todo el código máquina necesario para ejecutar el programa con sentencias DATA, y puede ser entrado y ejecutado tal como está. Damos también un listado del código fuente. Si desea digitarlo y ensamblarlo, puede omitir las líneas que van de 1520 a 1540 y las sentencias DATA entre 1620 y 1940.

En medio de la ejecución, intente obtener el modelo que le presentamos de un ritmo clásico:

```
        ****GENERADOR DE RITMOS DEL C64****
TEMPO=15
X _ _ _ X _ _ X X _ _ _ X _ _ _   BAJO
_ _ X _ _ _ X _ _ _ X _ _ _ X X   TIMBAL
_ X _ X _ X _ X _ X _ X _ X _ X   CENCERRO
```

Programa BASIC generador de ritmos

```
1000 REM ** GENERADOR DE RITMOS **
1010 PRINTCHR$(147):REM LIMPIA PANTALLA
1020 GOSUB1510:REM ESTABLECE C/M+SPRITE
1030 GOSUB1270:REM ESTABLECE PANTALLA
1040 SYS(49152):REM *INSERTA CUÑA*
1050 REM SYS (49175) PARA QUITAR CUÑA
1060 REM *BUCLE PRINCIPAL*
1070 P=PEEK(197):REM TECLA PULSADA
1080 IFP=37THENY=Y-16:SY=SY-1:IFY<95THENY=95:SY=0
1090 IFP=36THENY=Y+16:SY=SY+1:IFY>127THENY=127:SY=2
1100 IFP=47THENX=X-16:SX=SX-1:IFX<28THENX=28:SX=0
1110 IFP=44THENX=X+16:SX=SX+1:IFX>268THENX=268:SX=15
1120 Z=0:IFP=1 THENZ=1
1130 PRINTCHR$(19);S$;"TEMPO=";PEEK(679);CHR$(157);"      "
1140 POKEVIC+16,X/256:POKEVIC,XAND255
1150 POKEVIC+1,Y
1160 REM**CALCULO SITUACION PANTALLA POR LA POS X,Y DEL SPRITE**
1170 SC=B+INT((Y-50)/8+1)*40+INT((X-18)/8)
1180 IFSY=0THENROW=50000:REM RITMO DE BAJO
1190 IFSY=1THENROW=50016:REM RITMO DE TIMBAL
1200 IFSY=2THENROW=50032:REM CENCERRO
1210 IFZ=1ANDPEEK(SC)=32THENPOKESC,24:POKEROW+SX,1:GOTO1230
1220 IFZ=1ANDPEEK(SC)=24THENPOKESC,32:POKEROW+SX,0
1230 IFP=51THENGOSUB1270:FORI=0TO50:POKE50000+I,0:NEXT
1240 IFP=57THENPOKE198,0:POKEVIC+21,0:PRINTCHR$(147):END
1250 GOTO1070
1260 REM *ESTABLECE PANTALLA*
1270 PRINTCHR$(19);
1280 S$=CHR$(17)+CHR$(17)+CHR$(17)
1290 PRINTTAB(3)"      "
1300 PRINTCHR$(18)TAB(3)" **** GENERADOR RITMOS COMM64 ****"
1310 PRINT:PRINT:PRINT
1320 PRINT" ---------------------------------------
1330 PRINT"                                          -BAJO"

1340 PRINT" --------------------------------------- "
1350 PRINT"                                          -TIMBAL"
     "
1360 PRINT" --------------------------------------- "
1370 PRINT"                                          -CENCERRO"

1380 PRINT" --------------------------------------- "
1390 PRINT:PRINTTAB(8)          "OPCIONES...."
1400 PRINTTAB(8)"[F1]           DAR RITMO"
1410 PRINTTAB(8)"[F3]           REDUCIR TEMPO"
1420 PRINTTAB(8)"[F5]           AUMENTAR TEMPO"
1430 PRINTTAB(8)"[F7]           PARAR RITMO"
1440 PRINTTAB(8)"[CLR]          BORRAR DIAGRAMA"
1450 PRINTTAB(8)"[8T]           FINALIZAR PROGRAMA"
1460 PRINTTAB(8)"[<]            MOVER CURSOR IZQ"
1470 PRINTTAB(8)"[>]            MOVER CURSOR DERECHA"
1480 PRINTTAB(8)"[K]            SUBIR CURSOR"
1490 PRINTTAB(8)"[M]            BAJAR CURSOR"
1500 RETURN
1510 REM *ESTABLECE COD MAQ Y SPRITE*
1520 FORI=49152TO49413
1530 READJ:C=C+J:POKEI,J:NEXTI
1540 READJ:IFC<>JTHENPRINT"ERROR DATOS":END
1550 FORI=0TO62:READJ:POKE832+I,J:NEXTI
1560 VIC=53248:X=28:Y=95:B=1024:SID=54272
1570 FORI=0TO24:POKESID+I,0:NEXTI
1580 POKESID+24,15:POKEVIC+21,1:POKE2040,13
1590 POKE254,15:POKE679,15:POKEVIC+39,1
1600 FORI=0TO50:POKE50000+I,0:NEXT
1610 RETURN
1620 REM***DATOS M/C***
1630 DATA120,173,20,3,133,251,173,21,3
1640 DATA133,252,169,36,141,20,3,169
1650 DATA192,141,21,3,88,96,120,165,251
1660 DATA141,20,3,165,252,141,21,3,88
1670 DATA96,32,177,192,165,197,201,3
1680 DATA208,3,32,135,192,165,197,201,4
1690 DATA208,3,32,140,192,165,197,201,5
1700 DATA208,3,32,155,192,165,197,201,6
1710 DATA208,3,32,166,192,32,87,192,142
1720 DATA168,2,140,169,2,76,49,234,165
1730 DATA253,208,1,96,198,254,240,1,96
1740 DATA32,149,192,224,16,208,3,32,144
1750 DATA192,185,80,195,240,3,32,184
1760 DATA192,185,96,195,240,3,32,210
1770 DATA192,185,112,195,240,3,32,236
1780 DATA192,200,234,232,96,169,0,133
1790 DATA253,96,169,1,133,253,162,0,160
1800 DATA0,96,173,167,2,133,254,96,173
1810 DATA167,2,201,255,240,3,238,167,2
1820 DATA96,173,167,2,201,1,240,3,206
1830 DATA167,2,96,174,168,2,172,169,2
1840 DATA96,169,14,141,6,212,169,32,141
1850 DATA2,212,169,66,141,4,212,169,3
1860 DATA141,1,212,169,65,141,4,212,96
1870 DATA169,7,141,12,212,169,12,141,13
1880 DATA212,169,128,141,11,212,169,65
1890 DATA141,8,212,169,129,141,11,212
1900 DATA96,169,2,141,19,212,169,13,141
1910 DATA20,212,169,18,141,18,212,169
1920 DATA100,141,15,212,169,17,141,18
1930 DATA212,96
1940 DATA32038:REM *SUMA DE CONTROL*
1950 REM**DATOS CURSOR SPRITE**
1960 DATA127,254,0,127,254,0,127,254,0
1970 DATA112,14,0,112,14,0,112,14,0,112
1980 DATA14,0,112,14,0,112,14,0,112,14
1990 DATA0,112,14,0,127,254,0,127,254,0
2000 DATA127,254,0,0,0,0,0,0,0,0,0,0,0
2010 DATA,0,0,0,0,0,0,0,0,0,0
```

Listado assembly

```
; ++++++++++++++++
; ++++++++++++++++
; ++                ++
; ++   CODIGO FUENTE   ++
; ++ CAJA RITMOS CBM ++
; ++                ++
; ++++++++++++++++
; ++++++++++++++++
;
VOL     =  $D418   ;VOLUMEN DEL SID
ATT1    =  $D405   ;SUBIDA VOZ 1
SUS1    =  $D406   ;RETENCION VOZ 1
PULSE   =  $D402   ;VELOCIDAD PULSACIONES VOZ 1
WAVE1   =  $D404   ;FORMA ONDA VOZ 1
BASS    =  $D401   ;FRECUENCIA BYTE HI VOZ 1
ATT2    =  $D40C   ;SUBIDA VOZ 2
SUS2    =  $D40D   ;RETENCION VOZ 2
WAVE2   =  $D40B   ;FORMA ONDA VOZ 2
SNARE   =  $D408   ;FRECUENCIA BYTE HI VOZ 2
ATT3    =  $D413   ;SUBIDA VOZ 3
SUS3    =  $D414   ;RETENCION VOZ 3
WAVE3   =  $D412   ;FORMA ONDA VOZ 3
BELL    =  $D40F   ;FRECUENCIA BYTE HI VOZ 3
ROW1    =  $C350   ;ALMACENAMIENTO VOZ 1
ROW2    =  $C360   ;ALMACENAMIENTO VOZ 2
ROW3    =  $C370   ;ALMACENAMIENTO VOZ 3
TEMPO   =  $02A7   ;ALMACENAM. TEMPORAL DEL RETARDO
XCOUNT  =  $02A8   ;ALMACENAM. TEMP. DEL REGISTRO X
YCOUNT  =  $02A9   ;ALMACENAMIENTO TEMP. DEL REG Y
LOVEC   =  $FB     ;ALMACENAM. VECTOR BYTES LO
HIVEC   =  $FC     ;ALMACENAM VECTOR BYTES HI
PLAY    =  $FD     ;SONIDO TAMBOR (1=SI)
DELAY   =  $FE     ;ALMACENAM. ESTADO ACTUAL RETARDO
KEY     =  $C5     ;TECLA PULSADA
;-----------------------------------------------------------
*       =  $C000   ;ASSEMBLE DESDE LA 49152 (DECIMAL)
;-----------------------------------------------------------
; ESTABLECER CUÑA
;-----------------------------------------------------------
;
        SEI             ;DESACTIVA PETICION INTERRUPCIONES
        LDA $0314       ;TOMA VALORES DEL BYTE LO DEL VECTOR
        STA LOVEC       ;LOS ALMACENA EN LOVEC
        LDA $0315       ;TOMA VALORES DEL BYTE HI DEL VECTOR
        STA HIVEC       ;LOS ALMACENA EN HIVEC
        LDA #<WEDGE     ;TOMA BYTE LO DE DIR INICIO CUÑA
        STA $0314       ;LO ALMACENA EN BYTE LO DE VECTOR IRQ
        LDA #>WEDGE     ;TOMA BYTE HI DE DIR INICIO CUÑA
        STA $0315       ;LO ALMACENA EN BYTE HI DE VECTOR IRQ
        CLI             ;REANUDA PETICION INTERRUPCION
        RTS             ;RETURN
;-----------------------------------------------------------
; QUITAR CUÑA
;-----------------------------------------------------------
;
        SEI             ;DESACTIVA PETICION INTERRUPCIONES
        LDA LOVEC       ;TOMA VALOR ORIGINAL DE LOVEC
        STA $0314       ;LO ALMACENA EN BYTE LO DE VECTOR IRQ
        LDA HIVEC       ;TOMA VALOR ORIGINAL DE HIVEC
        STA $0315       ;LO ALMACENA EN BYTE HI DE VECTOR IRQ
        CLI             ;REANUDA PETICION INTERRUPCION
        RTS             ;RETURN
;-----------------------------------------------------------
; BUCLE PRINCIPAL
;-----------------------------------------------------------
WEDGE   JSR REG         ;GOSUB REG
        LDA KEY         ;QUE TECLA FUE PULSADA?
        CMP #$03        ;FUE LA TECLA FUNCION #1?
        BNE CONT1       ;SI NO, BIFURCAR
        JSR FLAG0       ;GOSUB FLAG0
CONT1   LDA KEY         ;QUE TECLA FUE PULSADA?
        CMP #$04        ;FUE LA TECLA FUNCION #7?
        BNE CONT2       ;SI NO, BIFURCAR
        JSR FLAG1       ;GOSUB FLAG1
CONT2   LDA KEY         ;QUE TECLA FUE PULSADA?
        CMP #$05        ;FUE LA TECLA FUNCION #3?
        BNE CONT3       ;SI NO, BIFURCAR
        JSR ADD         ;GOSUB ADD
CONT3   LDA KEY         ;QUE TECLA FUE PULSADA?
        CMP #$06        ;FUE LA TECLA #5?
        BNE CONT4       ;SI NO, BIFURCAR
        JSR MINUS       ;GOSUB MINUS
CONT4   JSR REST        ;GOSUB REST
        STX XCOUNT      ;ALMACENA VALOR REGISTRO X
        STY YCOUNT      ;ALMACENA VALOR REGISTRO Y
        JMP $EA31       ;VUELVE A INTERR
;-----------------------------------------------------------
; RUTINA PRUEBA
;-----------------------------------------------------------
REST    LDA PLAY        ;TOMA EL VALOR DEL TOGGLE
        BNE BEGIN       ;SI ES 1 BIFURCAR
        RTS             ;RETURN
;-----------------------------------------------------------
BEGIN   DEC DELAY       ;DECREMENTA EL RETARDO
        BEQ START       ;BIFURCAR SI ES CERO
        RTS             ;RETURN
;-----------------------------------------------------------
START   JSR COUNT       ;CONTADOR GOSUB
        CPX #$10        ;FIN DEL BUCLE?
        BNE CHECK       ;SI NO ES 0, BIFURCAR
        JSR RESET       ;RESTAURA GOSUB
;-----------------------------------------------------------
CHECK   LDA ROW1,Y      ;TOMA VALOR DE DESP. FILA 1 POR Y
        BEQ NEXT1       ;SI ES CERO, BIFURCAR
        JSR DRUM1       ;GOSUB DRUM 1
NEXT1   LDA ROW2,Y      ;TOMA VALOR DE DESP. FILA 2 POR Y
        BEQ NEXT2       ;SI ES CERO, BIFURCAR
        JSR DRUM2       ;GOSUB DRUM2
NEXT2   LDA ROW3,Y      ;TOMA VALOR DE DESP. FILA 3 POR Y
        BEQ NEXT3       ;SI ES CERO, BIFURCAR
        JSR DRUM3       ;GOSUB DRUM3
NEXT3   INY             ;INCREMENTA DESPLAZAMIENTO
        INX             ;INCREMENTA CONTADOR BUCLE
        RTS             ;RETURN
;-----------------------------------------------------------
; SUBRUTINAS
;-----------------------------------------------------------
FLAG0   LDA #$00        ;ALMACENA CERO -
        STA PLAY        ;INTRODUCE PLAY
        RTS             ;RETURN
;-----------------------------------------------------------
FLAG1   LDA #$01        ;ALMACENA 1 -
        STA PLAY        ;INTRODUCE PLAY
RESET   LDX #$00        ;RESTAURA REG X
        LDY #$00        ;RESTAURA REG Y
        RTS             ;RETURN
;-----------------------------------------------------------
COUNT   LDA TEMPO       ;TOMA VALOR TEMPO
        STA DELAY       ;ALMACENA RETARDO
        RTS             ;RETURN
;-----------------------------------------------------------
ADD     LDA TEMPO       ;TOMA VALOR TEMPO
        CMP #$FF        ;COMPARA RESULTADO CON 255
        BEQ CONT5       ;SI NO ES 255, BIFURCAR
        INC TEMPO       ;INCREMENTA TEMPO
CONT5   RTS             ;RETURN
;-----------------------------------------------------------
MINUS   LDA TEMPO       ;TOMA VALOR TEMPO
        CMP #$01        ;COMPARA RESULTADO CON 1
        BEQ CONT6       ;SI NO ES 1, BIFURCAR
        DEC TEMPO       ;DECREMENTA TEMPO
CONT6   RTS             ;RETURN
;-----------------------------------------------------------
REG     LDX XCOUNT      ;ALMACENA VALOR CONT X EN REG X
        LDY YCOUNT      ;ALMACENA VALOR CONT Y EN REG Y
        RTS             ;RETURN
;-----------------------------------------------------------
; RUTINAS TOQUE DE TAMBOR
;-----------------------------------------------------------
DRUM1   LDA #$0E        ;ESTA-
        STA SUS1        ;
        LDA #$20        ;BLECER
        STA PULSE       ;
        LDA #$42        ;TAMBOR
        STA WAVE1       ;
        LDA #$03        ;BAJO
        STA BASS        ;
        LDA #$41        ;Y TOCAR
        STA WAVE1       ;
        RTS             ;VOLVER
;-----------------------------------------------------------
DRUM2   LDA #$07        ;ESTA-
        STA ATT2        ;
        LDA #$0C        ;BLECER
        STA SUS2        ;
        LDA #$80        ;TIMBAL
        STA WAVE2       ;
        LDA #$41        ;Y
        STA SNARE       ;
        LDA #$81        ;TOCAR
        STA WAVE2       ;
        RTS             ;VOLVER
;-----------------------------------------------------------
DRUM3   LDA #$02        ;ESTA-
        STA ATT3        ;
        LDA #$0D        ;BLECER
        STA SUS3        ;
        LDA #$12        ;CENCERRO
        STA WAVE3       ;
        LDA #$64        ;Y
        STA BELL        ;
        LDA #$11        ;TOCAR
        STA WAVE3       ;
        RTS             ;VOLVER
        .END
```

Editorial Delta, S.A.

mi computer



10086
9 788485 822836